Modeling and Simulation of Non-Isothermal Gas-Assisted Injection Molding for Non-Newtonian Fluids

Undergraduate Thesis

Presented in Partial Fulfillment of the Requirements for Graduation "with Research Distinction in Chemical and Biomolecular Engineering"

By

Eugenia L. Stanisauskis

*****

William G. Lowrie Department of Chemical and Biomolecular Engineering
The Ohio State University
Columbus, OH 43210

November 17, 2017

Thesis Committee:
Dr. Kurt W. Koelling, Advisor
Dr. Isamu Kusaka

Acknowledgments

I would like to thank all the people who have assisted me throughout the course of my research at Ohio State. I would like to thank my advisor Dr. Koelling for all his assistance and guidance during this process. I would like to thank Xutao Shi for his help and advice during the course of my work as well. I would like to thank Dr. Kusaka for serving on my committee. Additionally, I would like to thank my mother and sister for their constant support and encouragement. Finally, I would like to thank The Ohio State University Undergraduate Research Scholarship Fund for the financial assistance they provided me with.

Abstract

Gas-assisted injection molding (GAIM) is a process where a molten polymer is injected into a cooled mold cavity followed by an injection of gas, allowing the polymer to fill the cavity and form a hollow part as it cools and solidifies. Current GAIM simulations for non-isothermal Newtonian fluids can accurately predict the coating thickness of the polymer melt. However, no existing models predict the behavior of non-isothermal non-Newtonian fluids. The proposed simulations use known material and system properties and process variables such as mold temperature, melt temperature, and pressure, to calculate the temperature and velocity profiles to predict the coating thickness using derived relations. Experiments have been performed to determine the effect of temperature gradients and rheological behavior of the penetrating fluid. The experimental data was compared against the simulation for both Newtonian and shear-thinning fluids utilizing various viscosity models to describe the behavior of the fluids and showed that the simulations were reliable in predicting the fractional coverages at higher rowlengths. Velocity and temperature profiles were calculated in which the non-isothermal behavior of the system was shown to be predicted by the simulation. In addition, viscosity vs shear rate plots were created to validate the assumed behavior of the tested fluids for more accurate modeling. Simulated vs experimental results are plotted for the cross-exponential viscosity model for both polycarbonate and polystyrene at various rowlengths, and additionally the Ellis and power law viscosity models for polystyrene. An accurate simulation will help to reduce production costs and improve product quality.

Vita

2013.....Bishop Gorman High School

2017.....B.S. Chemical and Biomolecular Engineering, The Ohio State University

Field of Study

Major Filed: Chemical and Biomolecular Engineering

Contents

List of Figures

List of Tables

1 Introduction

Gas-assisted injection molding is a process involving the penetration of an inviscid gas through a viscous, non-Newtonian polymer under non-isothermal conditions. The process begins with the injection of a polymer melt into a hollow mold. After a certain amount of time, termed the delay time, a gas is inserted into the mold. This injection produces a bubble that forces the polymer melt against the wall. Once the polymer melt has cooled and solidified, the result is a hollow part that is light and relatively cheap to make compared to other conventional methods. An image of this process can be seen in Figure 1.

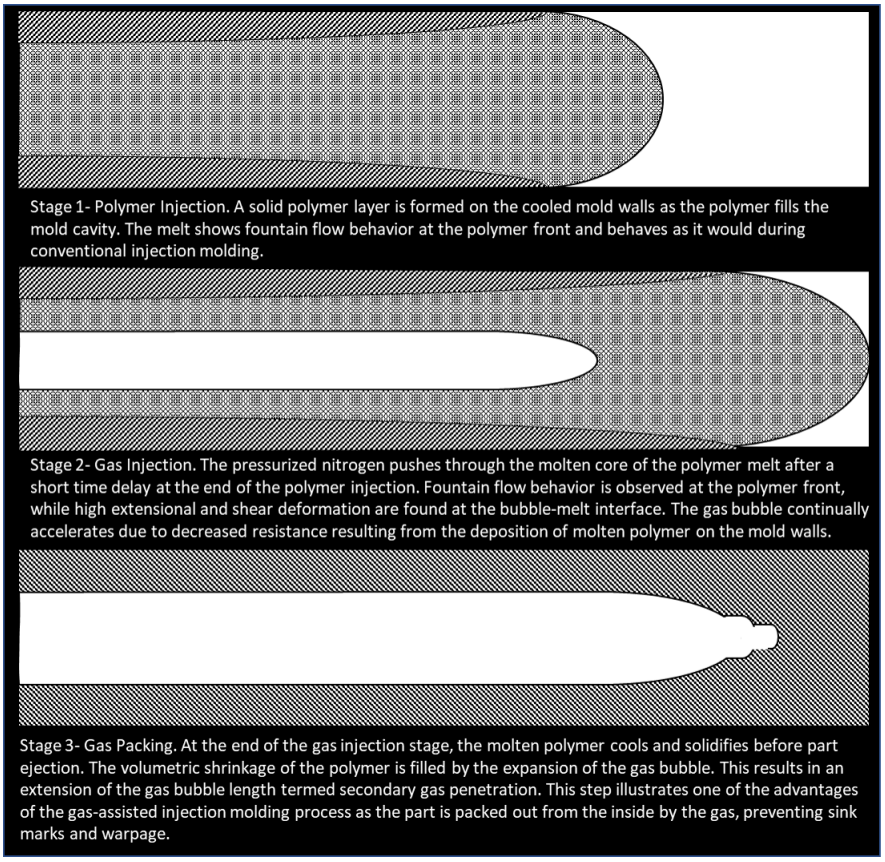


Figure 1: The Three Stages of the Gas-Assisted Injection Molding Process.

GAIM was developed to overcome the limitations of conventional injection molding. The GAIM process is able to produce plastic parts that are up to 30% lighter, have higher strength to weight ratios, are stiffer, and show ascetic improvements when compared to

conventional injection molded parts. Since the process consists of an incomplete injection of molten polymer, or short shot, followed by an injection of compressed nitrogen gas, a hollow part that is up to 50% gas, by volume, is produced. Parts molded by conventional injection molding technology are packed out through high screw holding pressures, but can only be packed if the gates to the mold cavity are not frozen. A gas injection molded part, however, is packed from within by the inert, high pressure gas throughout the cooling cycle producing better dimensional stability.

GAIM provides further benefits as well. The gas-assist technique is ideal for adding thick, hollowed-out sections to otherwise thin-walled parts and creating these thin-walled parts with low clamp tonnage, reducing both tooling cost and required injection molding machine size.[7] These advantages allow for functional, economical, and manufacturing benefits. Parts made by GAIM are commonly used in the plastics processing, and medical device industries.

Simulating the behavior of GAIM processes is necessary when attempting to control mold design and obtain a good quality product. GAIM simulation software is available but all these softwares are based on 3D models. The detriments to these models are that full-scale 3D modeling requires discretization of thin parts resulting in unreasonable computation times and high calculation instabilities. Previous work has been done in which alternative simulation models were created to predict coating thickness of non-Newtonian fluids in a free convection setting.

This work expanded upon the model for non-Newtonian fluids through the incorporation of the Cross-Exponential, Ellis, and Power Law models. For the Ellis and Cross-Exponential models, the modeling is based on the use of Mathematica to solve non-linear ordinary differential equations, as well as the use of dimensionless variables. The Power Law model utilizes FDM in its solution. In addition, this work compared the simulated results with experimental data based on varying process conditions in order to analyze the accuracy of the model at each set of conditions. The simulation results are compared with experimental data collected by Kaminski[9] for verification.

2 Literature Review

2.1 Newtonian Isothemal Gas-Liquid Modeling Experiments

Previous work has been done in which the behavior of GAIM processes under isothermal systems have been studied. Fairbrother conducted the first experiments in this area with viscous Newtonian fluids[5]. His studies determined a relationship between capillary number (Ca) and fractional coverage (m). Fractional coverage is the amount of fluid left behind on the mold after the gas bubble has penetrated through. Fairbrother was able to develop an equation relating fractional coverage and capillary number. This relation can be seen in Equation 1.

$$m = Ca^{0.5} \tag{1}$$

This equation holds true for capillary numbers up to 0.009. An equation for fractional coverage in tube shaped geometries can be seen in Equation 2.

$$m = \frac{A_p}{A_t} = 1 - \frac{R_x}{R_t}^2 \tag{2}$$

In this equation, A_p is the cross-sectional area of the polymer deposited on the tube walls by the bubble, A_t is the cross-sectional area of the entire tube, R_x is the radius of the gas bubble, and R_t is the radius of the tube. Capillary number is the ratio of viscous to surface tension forces acting across an interface between a liquid and a gas. A relation for capillary number can be found in Equation 3.

$$Ca = \frac{\eta U_b}{\sigma} \tag{3}$$

where η represents the viscosity of the fluid, U_b is the velocity of the bubble, and σ is the fluid surface tension.

Taylor went on to develop this investigation to extend the range of capillary numbers up to 2.0[14]. The next year, Cox went on to to extend the research to higher capillary numbers and determined that fractional coverage reached an asymptotic value of m = 0.6 for Newtonian fluids with capillary numbers greater than 10[4].

2.2 Non-Newtonian Isothermal Gas-Liquid Modeling Experiments

Poslinski and Coyle went on to conduct similar isothermal experiments using silicone liquid pastes. These pastes exhibited Bingham behavior, in which the fluids displayed shear-thinning behavior at low shear rates and a Newtonian plateau at higher shear rates[11][12]. Poslinski and Coyle used an isothermal computer simulation to model the penetration of a constant velocity gas bubble through an isothermal, shear-thinning fluid[12]. Their results showed good correlations with their experimental results. Huyzak and Koelling went on to study the same system with Boger fluids[8]. The Boger fluids showed fractional coverages that matched the Newtonian fluid relations at low capillary numbers, but increased due to higher extension rates at the bubble tip above a capillary number of six. The fractional coverage ultimately asymptoped to 0.75 around $Ca \approx 200$. The higher fractional coverages were determined to be due to the elastic properties of the Boger fluids, while shear-thinning behavior produced lower asymptotic isothermal fractional coverages. Gauri and Koelling extended the study of long gas bubbles in viscoelastic fluids in capillary tubes[6]. Kaminski went on to throughly study the effects of multiple process parameters on the GAIM process [9]. Kaminski also developed numerical simulations of GAIM assuming isothermal conditions.

Yijie Wang then went to to develop models for Newtonian fluids exhibiting non-Newtonian behavior with a varying temperature gradient at high capillary numbers[15]. Chen improved numerical compuational times for existing simulation methods[2]. Shen studied 2D gas channel systems with penetrating gas bubbles[13]. Belblidia established correlations for the prediction of fractional coverages in a straight 2D cylindrical tube[1]. Li produced a surface model regarding the simulation of the filling process in GAIM[10]. A significant amount of experimentation and simulation has been completed for the GAIM process but simulations of non-Newtonian fluids under non-isothermal conditions at high capillary numbers have hardly been studied.

3 Methodology

3.1 Experimental Apparatus

The experimental work utilized a spiral mold cavity of 12.7 mm diameter with a flowlength of 585mm for two different transparent, amorphous, injection molding compounds: polystyrene (PS), and polycarbonate (PC). The properties of these materials will be discussed in Section 3.2. An image of the spiral mold cavity can be found in Figure 2. The mold itself was comprised of steel with embedded cooling channels that ran parallel to the mold cavity.

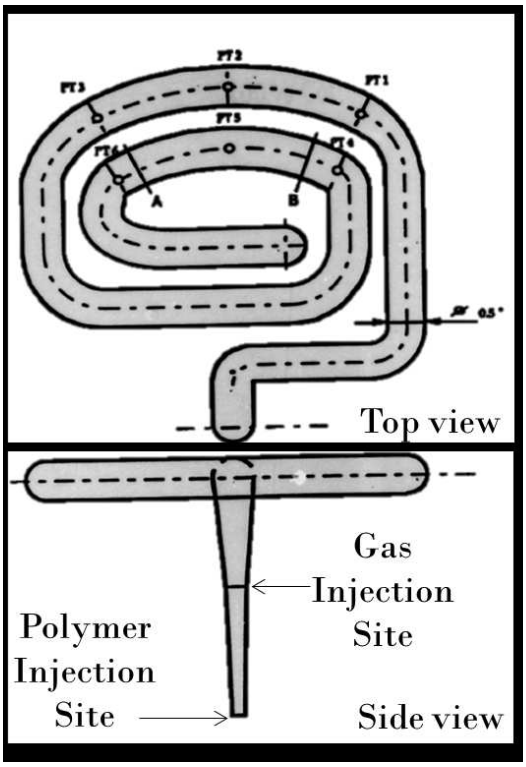


Figure 2: Spiral Mold Geometry Used in Gas-Assisted Injection Molding Trials.

Kaminski conducted experiments through a single factor matrix, in which each variable was changed to high and low settings, while keeping others constant. Table 1 lists the parameters varied at low, high, and base settings for the experiments. Each of the collected spiral samples were cross-sectioned and measured for wall thickness at different points along the flowlength. Six pressure transducers were used in order to determine the gas bubble

Table 1: Varied Parameters in Experiments for Each Material.

	PS			PC		
Parameter	**Low**	**Base**	**High**	**Low**	**Base**	**High**
Piston Speed $[\frac{in}{sec}]$	1.8	3.1	10.0	1.8	3.1	10.0
Delay Time [sec]	0.62	2.37	4.37	1.6	2.4	4.0
Pre-Charge Gas Pressure [psi]	200	350	500	175	325	500
Shot Size [inch]	2.85	3.0	3.15	2.9	3.0	3.1
Ram Speed $[\frac{in}{sec}]$	0.86	1.14	2.0	0.75	1.0	1.5
Mold Temperature [deg F]	95.6	115.6	136.6	-	180	-
Melt Temperature [deg F]	471.5	485.6	511.7	608.7	623.6	637.6

velocities and cooling times in the closed mold cavity and mounted along flush in the full spiral mold cavity at various locations. These locations can be seen in Figure 2. Video recording was also used to capture and analyze the movement of the gas and polymer fronts in the closed mold cavity.

3.2 Fluid Characterization

Two transparent injection molding grade compounds were studied: a general purpose polystyrene, Dow Styron 685 D (PS); a high viscosity polycarbonate, GE Lexan 101 (PC). Each compound exhibited unique rheological properties which were exploited to study the effects of their properties on the resulting fractional coverage of the molded test spirals. Table 2 lists various thermal properties of each material.

The fluids were characterized using the Cross-Exponential model. The shear rate dependent viscosity of each material at the base melt temperature can be seen in Figure 3. The Cross-exponential viscosity model is defined by Equation 4.

$$\eta = \frac{\eta_o}{1 + (\frac{\eta_o \dot{\gamma}}{\tau^*})^{1-n}} \tag{4}$$

Table 2: Thermal Properties of the Investigated Polymers.

Polymer	**Polystyrene**	**Polycarbonate**
Density $[\frac{kg}{m^3}]$	940	1017
Heat capacity $[\frac{J}{kgK}]$	2100	2052
Thermal conductivity $[\frac{W}{mK}]$	0.150	0.320
Thermal diffusivity $[\frac{m^2}{s}]$	$7.5x10^{-8}$	$1.53x10^{-7}$

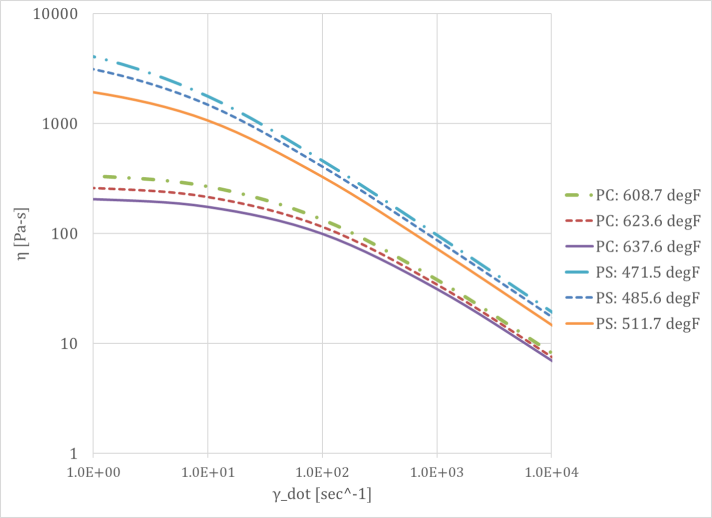


Figure 3: Shear Rate Dependent Viscosities of the Studied Polymers at the Tested Injection Temperatures.

where

$$\eta_o = Bexp\left(\frac{T_b}{T}\right), \tag{5}$$

η is the viscosity, T is the temperature, η_o is the zero shear-rate viscosity, $\dot{\gamma}$ is the shear rate, T_b is the temperature sensitivity coefficient, τ^* is the shear-stress level in relation to the transition between the Newtonian and shear-thinning regimes, and n is the shear thinning coefficient. It should be noted that the 1-n term in this equation is equal to the power law index. This relation will be utilized in the determination of asymptotic isothermal fractional coverage for non-Newtonian fluids. Table 3 lists values for the parameters of the Cross-Exponential model for both fluids.

Experimental work determined the tested shear rates to range from 20 s^{-1} to 4000 s^{-1}.

Table 3: Cross-Exponential Parameters for All Tested Fluids.

	Polystyrene	**Polycarbonate**
n	0.2968	0.03727
τ^* **[Pa]**	$1.738x10^4$	$9.4221x10^5$
B [Pa-s]	$1.6783x10^{-6}$	$1.0242x10^{-4}$
T_b **[K]**	$1.1376x10^4$	$9.0679x10^3$

Coupling this fact with Figure 3 can lead to the conclusion that PS demonstrated shear-thinning behavior throughout the entirety of the experimentation while PC has an upper Newtonian plateau that extended over a majority of the range, with the transition to shear-thinning behavior beginning around 600 s^{-1}. Therefore when modeling, PC was treated as a Newtonian fluid (this is additionally supported by the fact that the power law index found for PC was ≈ 1) and PS, non-Newtonian.

3.3 Modeling of Newtonian Fluids

While GAIM offers many advantages to conventional injection molding, there are many complexities in the system that must be taken into consideration. This section will focus on the modeling of Newtonian fluids through the calculation of the temperature profile, velocity profile, and fractional coverage. An image of expected temperature and velocity profiles for non-isothermal systems can be seen in Figure 4.

3.3.1 Calculation of Temperature Profile

A non-isothermal system leads to a temperature profile similar to the one seen in Figure 4. In a non-isothermal system, the temperature profile is greatest in the middle of the tube and lowest at the wall. The system is divided into three regions where heat transfer can occur: outside the tube, inside the tube, and inside the polymer melt. In relation to the experimental apparatus, "inside the polymer melt" would refer to inside the spiral mold cavity, "inside the tube" would refer to the wall of the mold cavity, and "outside the tube"

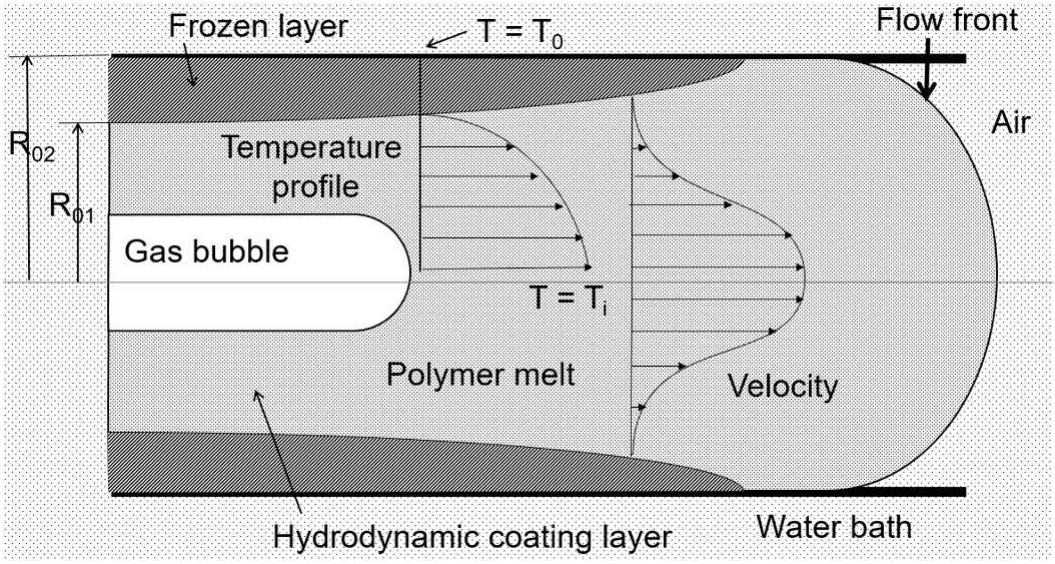


Figure 4: Bubble Penetration Through a Capillary Tube Filled with Polymer Melt Under Non-Isothermal Conditions.

would refer to the remaining portion of the steel mold.

The heat transfer outside the tube is a continuous process across the surrounding environment. This system is simplified by assuming the heat transfer occurs simply along the immediate surrounding area of the tube. Based off this assumption, the governing equation for heat transfer in this region is:

$$-k_s \frac{\partial T_s}{\partial r}\bigg|_{r=R_{out}} = h(T_w - T_B) \tag{6}$$

where k_s is the thermal conductivity of stainless steel, T_s is the temperature of the steel-water interface, R_{out} is the radius of the outer tube, h is the convective heat transfer coefficient of the assumed thin layer coating the tube, T_w is the temperature of the wall, and T_B is the temperature of the bulk fluid.

Finite difference method is then used to determine the temperature of the steel/water interface.

$$T_{s(N,j+1)} = T_{s(N,j)} + \frac{2\alpha_s dt}{dr_2^2} * \left[T_{s(N,j+1)} - T_{s(N,j)} - \left(\frac{R_{out} + \frac{dr_2}{2}}{R_{out}} \right) \left(\frac{h dr_2}{k_s} \right) (T_{s(N,j)} - T_B) \right] \tag{7}$$

where $T_{s(N,j+1)}$ is the temperature of the steel/water interface at time step j, dt is the specified

time step, dr_2 is the specified radial step change, and α_s is the thermal diffusivity of the steel.

The main type of heat transfer taken into consideration was forced convection due to the setup of the experimental apparatus. Due to this assumption, the convective heat transfer coefficient is determined by Equation 8[3].

$$\bar{Nu} = A * \frac{Re^{1/2} Pr^{1/3}}{\left[1 + \left(\frac{0.4}{Pr}\right)^{2/3}\right]^{1/4}} = \frac{2R_{out}\bar{h}}{k_f} \tag{8}$$

where $\bar{Nu}$ is the average Nusselt number, Re is the Reynolds number, Pr is the Prandtl number, A is a constant equal to 0.62, and k_f is the thermal conductivity of the bulk fluid. At this point in the simulation, velocity is not yet known making the Reynolds number calculation difficult. The influence of Reynold's number on the system was studied and found to have little to no impact on the temperature profile. Therefore, a representative value for Reynold's number was chosen for the simulation.

The heat transfer inside the tube is then taken into consideration. The governing equation for this region is

$$\frac{\partial T_s}{\partial t} = \alpha_s \left(\frac{\partial^2 T_s}{\partial r^2} + \frac{1}{r} \frac{\partial T_s}{\partial r} \right). \tag{9}$$

Again, through the implementation for FEM, Equation 9 becomes

$$T_{s(i,j+1)} = T_{s(i,j)} + \frac{\alpha_s dt}{r_{s(i)} dr_2^2} * \left[\left(\frac{r_{s(i)} + r_{s(i-1)}}{2} \right) T_{s(i-1,j)} - 2 r_{s(i)} T_{s(i,j)} + \left(\frac{r_{s(i+1)} + r_{s(i)}}{2} \right) T_{s(i+1,j)} \right] \tag{10}$$

where $T_{s(i,j)}$ is the temperature of the steel at node i at any time step j and $r_s(i)$ is the radial position in the steel at node i.

The following initial and boundary conditions were applied:

$$IC1 : @t = 0, T_{s(i,0)} = T_i \tag{11}$$

where T_i is the initial temperature of polymer melt.

$$BC1 : @r = R_{out}, T_{s(N,j)} = T_c \tag{12}$$

where T_c is the initial temperature of the cooling fluid.

$$BC2 : @r = R_{in}, q_{cond(n-1\rightarrow n)}A_1 + q_{cond(n+1\rightarrow n)}A_2 = q_{stored(n)}A_3 \tag{13}$$

FEM is applied on the last boundary condition to obtain

$$\begin{aligned} T_{s(i,j+1)} = T_{s(i,j)} + \frac{2k_s dt(R_{in} + \frac{dr_2}{2})(T_{s(i+1,j)} - T_{s(i,j)})}{dr_2[(R_{in} + \frac{dr_2}{4})\rho_s C_{p,s} dr_2 + (R_{in} - \frac{dr_1}{4})\rho_s C_{p,p} dr_1]} \\ + \frac{2k_p dt(R_{in} - \frac{dr_1}{2})(T_{s(N-1,j)} - T_{s(i,j)})}{dr_1[(R_{in} + \frac{dr_2}{4})\rho_s C_{p,s} dr_2 + (R_{in} - \frac{dr_1}{4})\rho_s C_{p,p} dr_1]} \end{aligned} \tag{14}$$

where dr_1 is the radial step change in the polymer.

The last region in which heat transfer is considered is region inside the polymer. The governing equation for this region can be found in Equation 15.

$$\frac{\partial T_p}{\partial t} = \alpha_p \left(\frac{\partial^2 T_p}{\partial r^2} + \frac{1}{r}\frac{\partial T_p}{\partial r} \right) \tag{15}$$

where α_p is the thermal diffusivity of the polymer, and T_p is the temperature of the polymer. With the application of FEM, Equation 15 becomes

$$T_{p(i,j+1)} = T_{p(i,j)} + \frac{\alpha_p dt}{r_{p(i)} dr_1^2} * \left[\left(\frac{r_{p(i)} + r_{p(i-1)}}{2} \right) T_{p(i-1,j)} - 2r_{p(i)} T_{p(i,j)} + \left(\frac{r_{p(i+1)} + r_{p(i)}}{2} \right) T_{p(i+1,j)} \right] \tag{16}$$

where $r_{p(i)}$ is the radial position in the polymer at node i.

The applied initial and boundary conditions in this region were:

$$IC2 : @t = 0, T_{p(i,0)} = T_i \tag{17}$$

$$BC3 : @r = 0, -k_p \left. \frac{\partial T_p}{\partial r} \right|_{r=0} = 0 \tag{18}$$

Applying FEM to the last boundary condition, Equation 18 becomes

$$T_{p(i,j+1)} = T_{p(i,j)} + 4 \left(\frac{\alpha_p dt}{dr_1^2} \right) (T_{p(i+1,j)} - T_{p(i,j)}). \tag{19}$$

The implementation of these equations presumes that the calculated temperature profile is that of the region far in front of the penetrating bubble. In order to determine the temperature profile of the polymer near the bubble, it is assumed that as the bubble passes

through the polymer at high velocities, the fluid directly in front of the bubble is relocated between the region between the bubble and inner wall. Additionally, the region upon which the temperature profiles are changing from static flow to flow in front of the bubble are neglected for simplification. We assume the temperature profile in front of the bubble tip and far in front of the bubble are the same. Through the implementation of these assumptions and by calculating the radial position of the bubble curve at each node, the squeezed temperature profile is calculated. Temperature profiles for each time step in the system are determined as well.

3.3.2 Calculation of Velocity Profile

From Figure 4, it can be seen that the continuous temperature change in the system causes the velocity profile to deviate from the common parabolic shape expected for laminar flow in a pipe for Newtonian fluids. To begin the calculation of the velocity profile, pseudo-steady state is assumed. The momentum balance for flow in a tube was used to solve for the velocity profile.

$$\frac{\partial}{\partial r}\left(r\eta\frac{\partial u_z}{\partial r}\right) = \frac{\Delta P r}{L} \tag{20}$$

where ΔP is the pressure gradient and L is the tube length. From Equation 20, it can be observed that this equation has no temperature dependence. To account for the temperature dependence in our system, Equation 5 is coupled with the momentum balance. Recall that the zero shear viscosity is the viscosity measured for shear deformation at low shear rates. This fact coupled with the fact that Newtonian behavior was observed for all materials at low shear rates supports the coupling of Equation 5 with Equation 20. Therefore, Equation 20 becomes

$$\frac{\partial}{\partial r}\left(rB * exp\left(\frac{T_b}{T}\right)\frac{\partial u_z}{\partial r}\right) = \frac{\Delta P r}{L}. \tag{21}$$

Applying FEM on Equation 21 leads to

$$u_{z(i,j)} = u_{z(i-1,j)} + \frac{1}{2(N-1)}\left(\frac{\Delta P r_{p(i)}}{2LB}\right) * \left[\frac{r_{p(i)}}{exp(\frac{T_b}{T_{p(i,j)}})} + \frac{r_{p(i-1,j)}}{exp(\frac{T_b}{T_{p(i,j)}})}\right]. \tag{22}$$

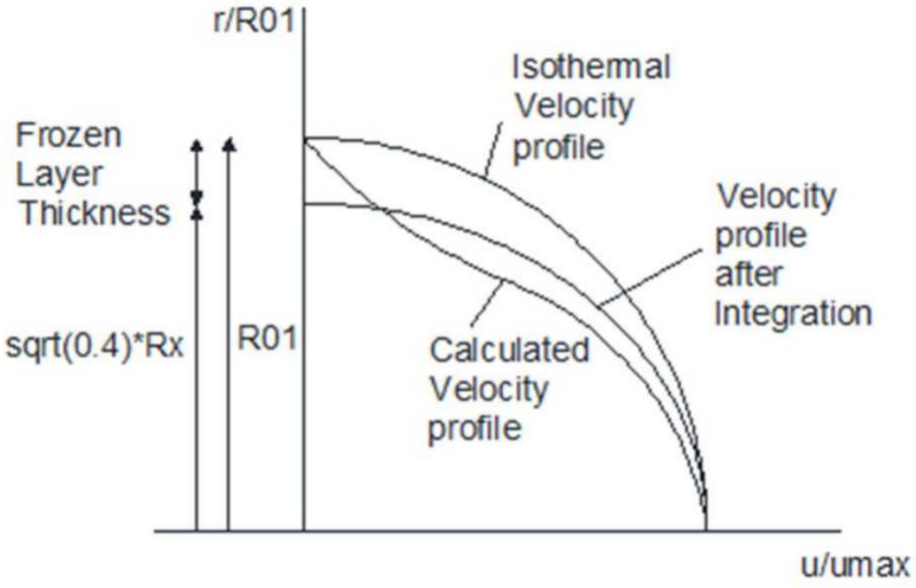


Figure 5: Isothermal, Calculated, and Assumed Velocity Profiles.[9]

The following boundary conditions were applied:

$$BC4 : u_z|_{r=R_{in}} = 0 \tag{23}$$

$$BC5 : \frac{du_z}{dr}\bigg|_{r=0} = 0 \tag{24}$$

3.3.3 Calculation of Fractional Coverage

To begin the calculation of the fractional coverage, one major assumption is made. This assumption is best depicted pictorially and can be seen in Figure 5. The velocity profiles of non-isothermal systems are not parabolic. This provides many complexities when modeling. In order to simplify our calculations, we assume the non-isothermal system is equal to an isothermal system with a frozen layer. The now parabolic shaped velocity profile can be describe by the equation:

$$u^* = 1 - \left(\frac{r}{R_x}\right)^2 \tag{25}$$

where u^* is the parabolic velocity profile and R_x is the radius of the bubble.

The volumetric flow rates for both the nonisothermal system and the isothermal system

with the frozen layer are assumed to be equal. This can be described by Equation 26.

$$2\pi \int_0^{R_{in}} r u_z^* dr = 2\pi \int_0^{R_x} r u^* dr \tag{26}$$

This equation can be integrated and solved to determine R_x. Once R_x is calculated, fractional coverage can be determined with the following equation:

$$m = 1 - (1 - m_\infty)\left(\frac{R_x}{R_{in}}\right)^2 \tag{27}$$

where m_∞ is the asymptotic isothermal fractional coverage. For Newtonian fluids at high capillary numbers, this value has been found to be 0.6[4]. This method was implemented for the determination of fractional coverage for polycarbonate. The written MATLAB code can be found in Appendix B. The results from this implementation are discussed in the next chapter.

3.4 Modeling of Non-Newtonian Fluids

When determining the temperature for the system with non-Newtonian fluids, the calculations provided in the previous section are still valid. Variations arise when determining a proper model for the viscosity of the fluid. When modeling non-Newtonian fluids, three viscosity models were applied: the cross-exponential model, the Ellis model, and the power law model. Their implementations are described in this section.

3.4.1 Use of Cross-Exponential Model

For non-Newtonian fluids, the assumption of Equation 5 to model the viscosity is no longer valid due to the shear-thinning behavior that occurs at higher shear rates. Therefore, Equation 4 was used. Combining Equations 4 and 20 gives us

$$\frac{\partial}{\partial r}\left(r \frac{\eta_o}{1 + (\frac{\eta_o \dot{\gamma}}{\tau^*})^{1-n}} \dot{\gamma}\right) = \frac{\Delta P r}{L}. \tag{28}$$

To employ FEM on this equation, $\dot{\gamma}$ must be solved for explicitly. However, due to the nonlinearity of this equation, no analytical solution can be determined. Therefore, this equation was solved numerically.

When solving this equation, it is convenient to introduce the following dimensionless variables:

$$v^* = \frac{\Delta P R_{in}^2}{L\eta_o} \tag{29}$$

$$r^* = \frac{r}{R_{in}} \tag{30}$$

$$E^* = \frac{\eta_o v^*}{\tau^* R_{in}} = \frac{\Delta P R_{in}^2}{\tau^* L} \tag{31}$$

$$\phi = \frac{u_z}{v^*} \tag{32}$$

$$\psi = \frac{d\phi}{dr^*} \tag{33}$$

Therefore, the Cross-exponential expression for the viscosity momentum balance becomes

$$\frac{d}{dr^*}\{r^*[1 + (E^*\psi)^{n-1}]\psi\} = r^*. \tag{34}$$

The boundary conditions of no-slip and symmetry about the pipe center line become as follows:

$$BC4 : \phi(1) = 0 \tag{35}$$

$$BC5 : \psi(0) = 0 \tag{36}$$

Although previous calculations had been completed in MATLAB, due to the nonlinearity of this ODE, Mathematica was used to solve this equation since most MATLAB functions are limited to solving linear ODEs. The code leading to the solution of this equation can be found in Appendix H.

It should be noted that when the ODE was made dimensionless, the only temperature dependent variable is ψ. This means that this variable is dependent on both temperature and radius. When redimensionalizing the variables, this dual dependence can be removed. The velocity profiles obtained from these calculations can be found in the next chapter.

To solve for the fractional coverage, again the assumption of equality of both a nonisothermal and isothermal system with a frozen layer is applied. When equating the volumetric flow rates, again is was found that equating the flowrates simplified the calculations. This

relation was found to be

$$\frac{Q_1^* L_1 \eta_{o,isothermal}}{R_{01}^4 \Delta P_1} = \frac{Q_x^* L_x \eta_{o,non-isothermal}}{R_x^4 \Delta P_x}. \tag{37}$$

where the subscript 1 refers to the non-isothermal system and x refers to the isothermal system. In this equation, it was assumed that the dimensionless flowrates and pressure gradients were equal. This simplified the equation and allowed for an analytical solution of R_x.

With a solution for R_x, fractional coverage was able to be solved with Equation 27. It should be noted that since the fluids considered are now non-Newtonian, the value for the asymptotic isothermal fractional coverage is not 0.6.

Poslinski and Coyle determined a relationship for fractional coverage based on the power law index[11]. This relationship was found to be

$$m_\infty = \begin{cases} -0.1516 ln(n) + 0.7259 & n \leq 0.3 \\ -0.1428n^2 + 0.2626n + 0.4782 & n > 0.3 \end{cases} \tag{38}$$

Fractional coverage was now able to be determined and results can be found in the next chapter. The code implemented for this solution can be found in Appendix D.

Since the cross-exponential model is significant only at low shear rates, the Ellis model was implemented for a more appropriate consideration of the shear-thinning behavior of non-Newtonian fluids.

3.4.2 Use of Ellis Model

The Ellis model can be expressed with the following equation

$$\eta = \frac{\eta_o}{1 + (\frac{\tau_{rz}}{\tau_{1/2}})^{\alpha - 1}} \tag{39}$$

where α is a measure of the degree of shear thinning behavior, $\tau_{1/2}$ is the value of shear stress at which the apparent viscosity has dropped to half its zero shear value, and τ_{rz} is the shear stress. The dependence of τ_{rz} in this equation allows for the consideration of the shear-thinning behavior of the modeled material.

Polystyrene was treated as a non-Newtonian fluid in these simulations. For this reason, the three Ellis model parameters were determined for this material. These parameters can be found in Table 4.

Table 4: Ellis Model Parameters for PS.

Parameter	**Value**
α	3.26
$\tau_{1/2}$ $[Pa - s^2]$	16618.6
η_o [Pa-s]	$1x10^{16}e^{-0.056T[K]}$

Due to the shear-thinning behavior of non-Newtonian fluids, τ_{rz} cannot be assumed to be constant. However, since laminar flow is assumed, the shear stress profile is known (see Figure 6). From this profile, an equation for τ_{rz} can be determined if shear stress at the wall

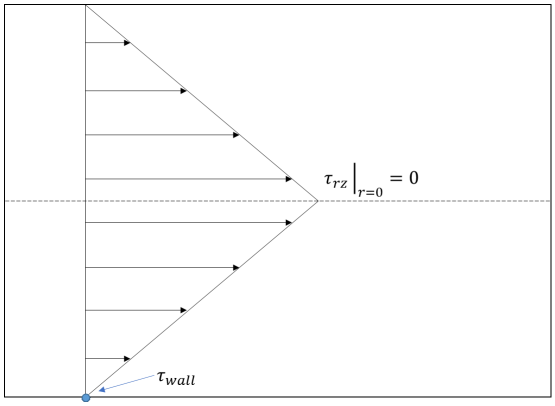


Figure 6: Shear Stress Profile for Laminar Flow in a Pipe.

is known. This relation can be found in Equation 40.

$$\tau_{rz} = \tau_{wall}\frac{r}{R_{01}} \tag{40}$$

With a relation for τ_{rz} now determined, the Ellis model can now be implemented into the momentum balance. Again, this ODE was made dimensionless for simplification purposes. The result is

$$\frac{d}{dr^*}\{r^*[1 + (e^*\Psi)^{\alpha-1}]\Psi\} = r^*. \tag{41}$$

where

$$e^* = \frac{\tau_{wall}}{\tau_{1/2}}, \tag{42}$$

$$\Psi = \frac{d\Phi}{dr^*}, \tag{43}$$

$$\Phi = \frac{u_z \nu_o}{2\tau_{wall}}. \tag{44}$$

Again, Mathematica was used to solve this ODE and the code reflecting this solution can be found in Appendix I. The dimensionless variables were used to solve for u_z and velocity profiles were determined.

Regarding the calculation of fractional coverage, the dimensionless flow rates were equated. This equation simplified to

$$R_x = R_{01}\sqrt{\frac{\tau_{wall,non-isothermal}\eta_{o,isothermal}}{\tau_{wall,isothermal}\eta_{o,non-isothermal}}}. \tag{45}$$

With R_x now determined, fractional coverage was calculated using Equations 27 and 38. The code implementing this solution is found in Appendix E and results can be found in the next chapter.

3.4.3 Use of Power Model

The power law model was also used to model the polystyrene. The power law model follows the form:

$$\eta = m\dot{\gamma}^{n-1} \tag{46}$$

where m and n are material dependent constants. The Power Law model was applied due to its simplicity during calculations.

With the coupling of the momentum balance and the viscosity equation, the following equation is reached:

$$\frac{\partial}{\partial r}\left(rm\dot{\gamma}^n\right) = \frac{\Delta Pr}{L} \tag{47}$$

The application of FEM to this equation leads to:

$$u_{z(i,j)} = u_{z(i-1,j)} + \frac{1}{2(N-1)}\left(\frac{\Delta Pr_{p(i)}}{2Lm}\right)^{\frac{1}{n}} * \left[\left(r_{p(i)}\right)^{\frac{1}{n}} + \left(r_{p(i-1,j)}\right)^{\frac{1}{n}}\right] \tag{48}$$

Previously stated boundary conditions were applied during the velocity profile calculations, Equation 26 was used to determine the bubble radius, and Equations 27 and 38. were used to calculate the fractional coverage.

4 Results

4.1 Simulation Results for Newtonian Fluids under Non-Isothermal Conditions

4.1.1 Temperature Profiles

Two temperature profiles are shown in this section to verify the non-isothermal behavior of the system. Figure 7 shows the calculated temperature profile for both the spiral mold cavity and the steel mold. It can be observed that very little heat transfer occurs through the steel mold due to its heat transfer properties and the difference in size between the spiral mold cavity and the steel mold.

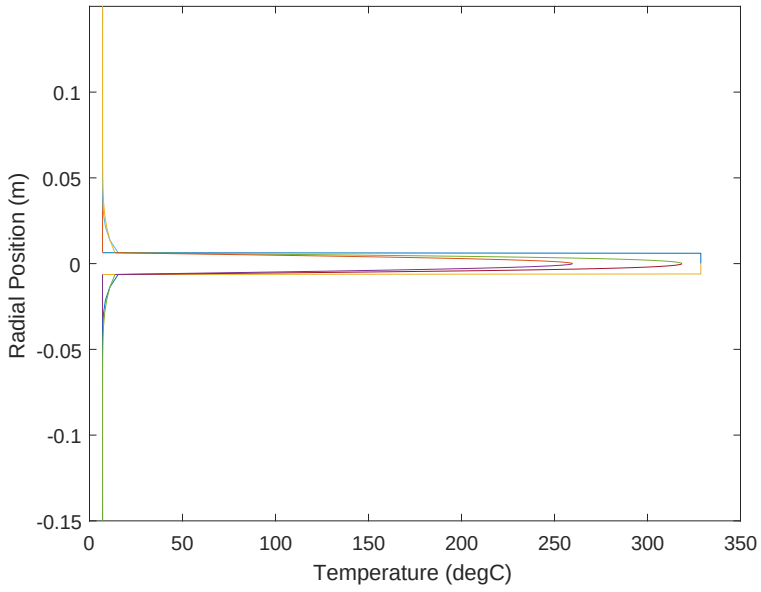


Figure 7: Temperature Profile for Both the Spiral Mold Cavity and Steel Mold for PC.

Figure 8 shows the calculated temperature profile for the spiral mold cavity. The parabolic nature of the profile as time increases confirms the non-isothermal assumption. In addition, the decreasing temperature as time increases confirms the phenomena of cooling as time increases.

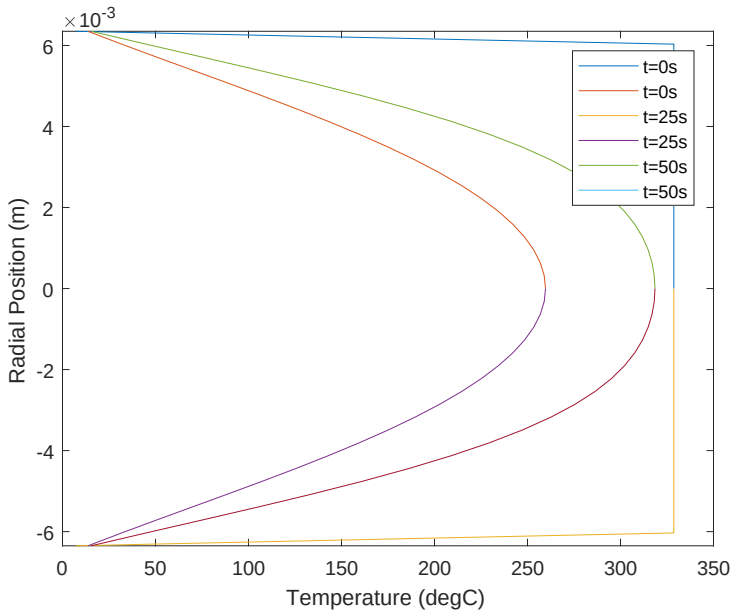


Figure 8: Temperature Profile for the Spiral Mold Cavity for PC.

4.1.2 Velocity Profiles

The normalized velocity profile was also determined for the spiral mold cavity. Figure 9 shows the calculated normalized velocity profile. Again, non-isothermal behavior is observed in the velocity profile as time goes on.

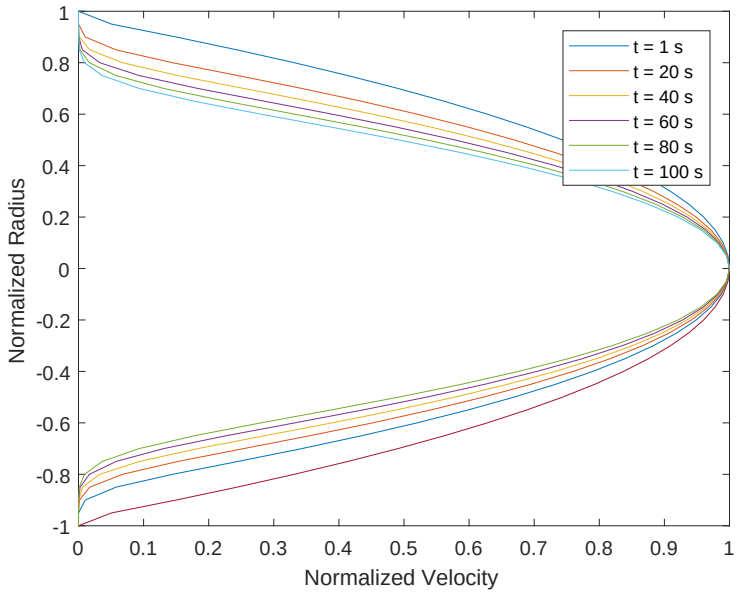


Figure 9: Normalized Velocity Profile for PC with Cross-Exponential Zero-Shear Viscosity Model.

4.1.3 m vs Fo at Base Conditions

With the calculated temperature and velocity profiles, the fractional coverage at each point was calculated and plotted against Fourier number to observe an estimate of the simulation vs the experimental data at all experiments. This simulation data may not necessarily be true for all points due to the changing conditions with each experiment. However, plotting all the points vs the simulation at base conditions allowed for an initial analysis of points to study in the system. Figure 10 shows the determined fractional coverage at base conditions using the Cross-exponential zero-shear viscosity model vs Fourier number. From the plot, it was determined to compare the simulated vs experimental fractional coverages at 90mm, 285mm, 343mm, and 505mm.

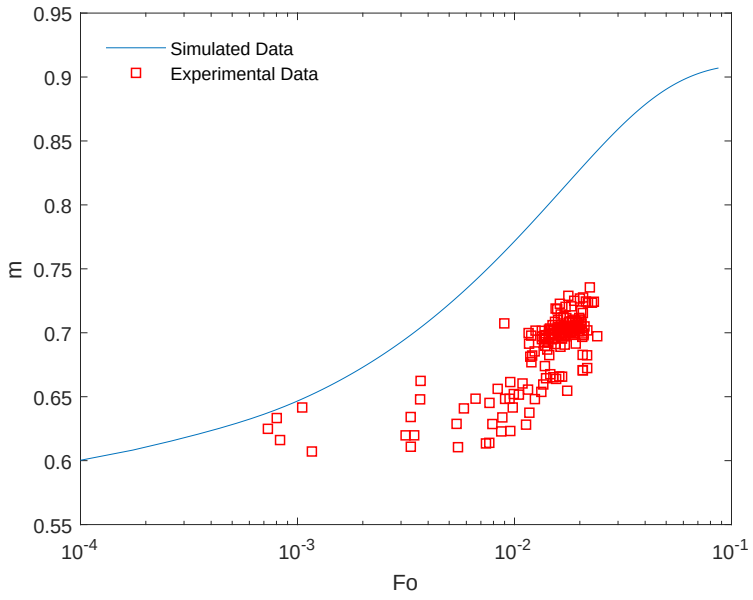


Figure 10: m vs Fo for Polycarbonate data with Cross-Exponential Zero-Shear Viscosity Model at Base Processing Conditions.

4.2 Comparison of Simulation and Experimental Results for Fractional Coverage of Newtonian Fluids

4.2.1 Effect of Shot Size

The first varying setting examined was shot size. In order to properly determine each point, first the viscosity and shear rate were determined to confirm the Newtonian behavior assumed. Figure 11 shows the shear rate vs viscosity for each point examined. The melt temperatures were approximately the same for each point causing the viscosity lines to lie very close together. Therefore, in order to easily see the difference between each simulated point, Figure 12 shows a zoomed in version of Figure 11. From these plots, we can conclude that all the examined points lie in the Newtonian region of the curve, making our assumption of Newtonian behavior valid.

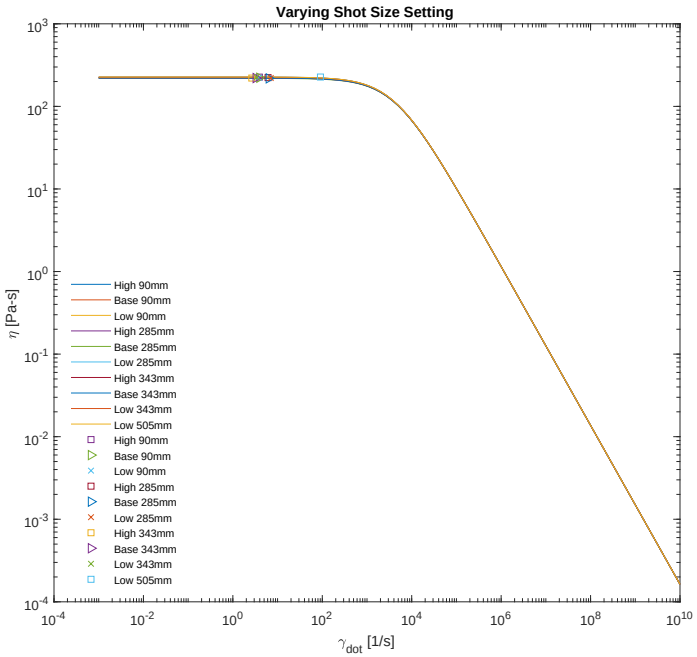


Figure 11: Viscosity vs Shear Rate for Each Simulated Point Varying Shot Size Setting for PC where High = 2.9in, Base = 3.0in, and Low = 3.1in.

With the validation of the assumption of Newtonian behavior, the fractional coverages

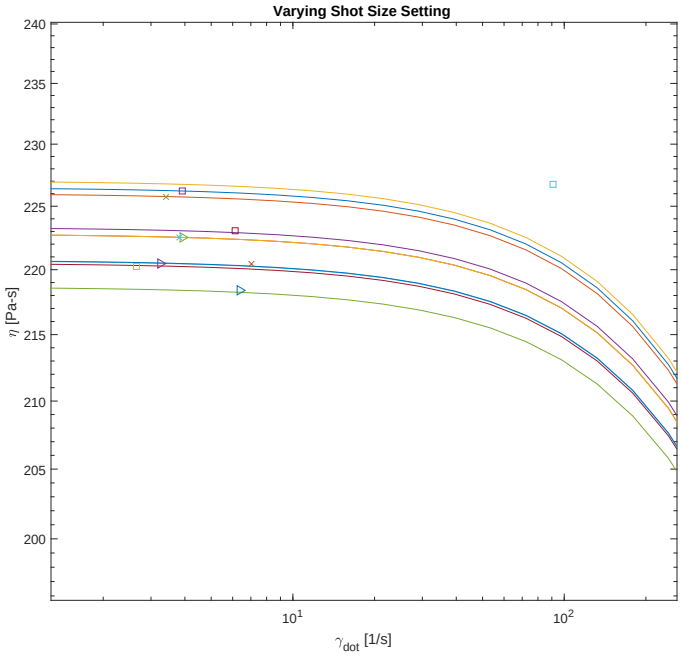


Figure 12: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Shot Size Setting for PC where High = 2.9in, Base = 3.0in, and Low = 3.1in.

were then compared for each point as seen in Figure 13. The simulation tends to predict fractional coverages that are higher than expected. For rowlengths 90mm, 285mm, and 343mm, the simulation tends to predict fractional coverages around 0.9. This prediction is more accurate for the high shot size setting although still higher than the experimental data. For 505mm, the point seems to be closer to the perfect simulation line. This could be due to the acceleration in velocity as the bubble travels down the tube which would increase the shear rate of the fluid making it slightly more non-Newtonian. However, the data points tend to be clustered together making these treads difficult to observe. No data was available for high and base shot size settings at 505mm due to the fact that for these systems, the bubble would not reach this rowlength based on the increase in shot size.

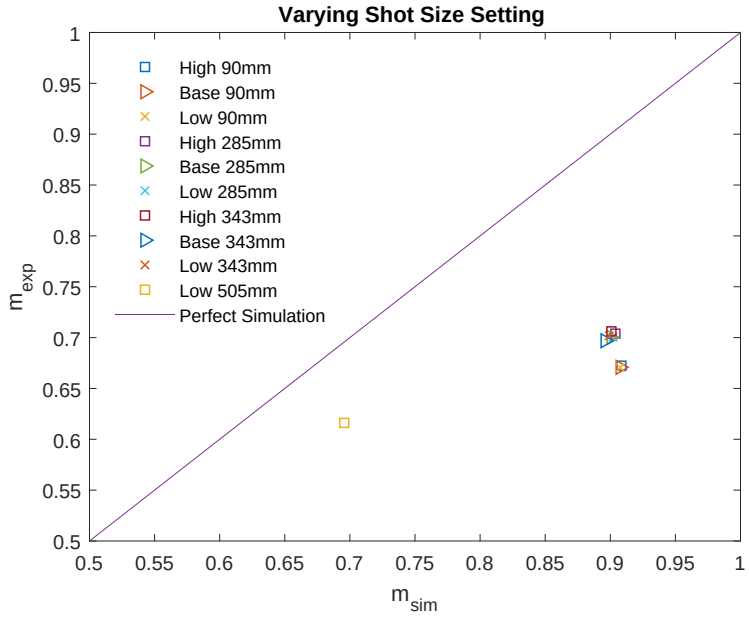


Figure 13: Simulated vs Experimental Fractional Coverage for PC with Varying Shot Size Setting where High = 2.9in, Base = 3.0in, and Low = 3.1in.

4.2.2 Effect of Delay Time

The next varied setting examined was delay time. Again shear rate vs viscosity was plotted for each simulation and Newtonian behavior was observed. Figure 14 shows the viscosity vs shear rate for each point and Figure 15 shows a zoomed in version of Figure 14.

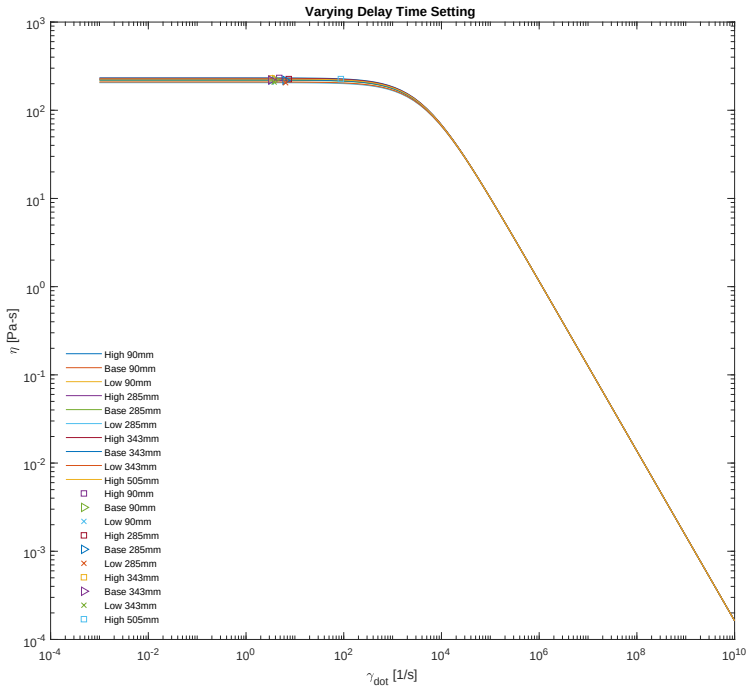


Figure 14: Viscosity vs Shear Rate for Each Simulated Point Varying Delay Time Setting for PC where High = 1.6sec, Base = 2.4sec, and High = 4.0sec.

Figure 16 shows the simulated vs experimental fractional coverage with varying delay time for multiple rowlengths. Unlike with the variation of shot size setting, these points are much less clustered, making trends slightly easier to observe. Again, the simulation tended to predict higher fractional coverages compared to the experimental data. However, we can observe that the simulation predicts more accurate fractional coverages for high delay time setting than the low setting. Also while m_{sim} at rowlengths 90mm, 285mm, and 343mm, tended to be much higher, at 505mm, the predicted fractional coverage is much closer to the experimental fractional coverage value.

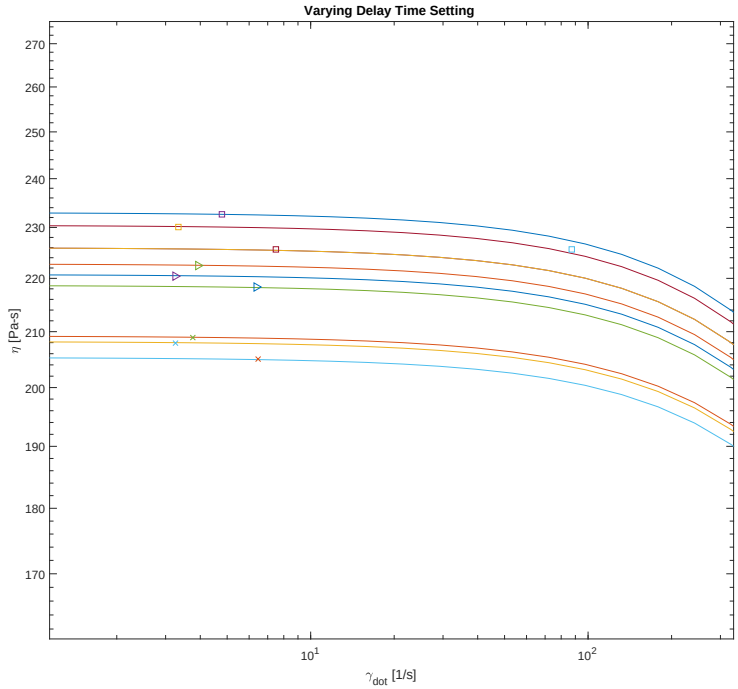


Figure 15: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Delay Time Setting for PC where High = 1.6sec, Base = 2.4sec, and High = 4.0sec.

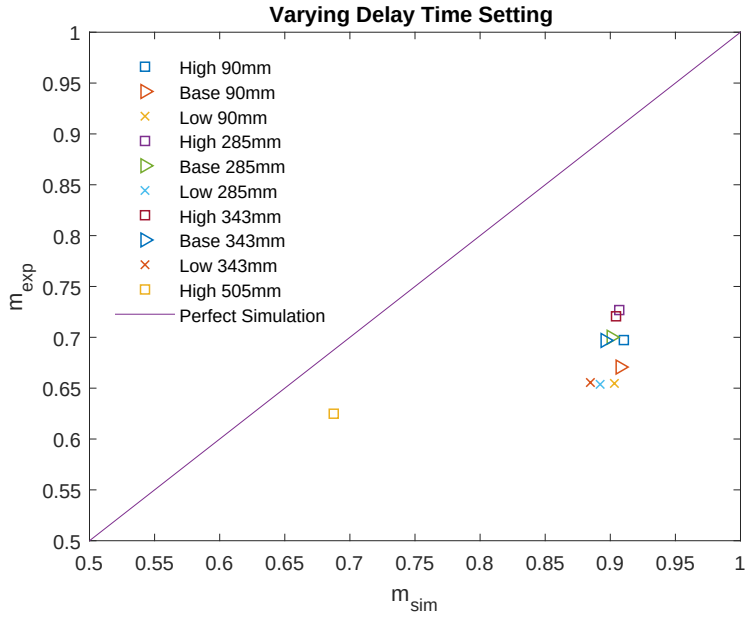


Figure 16: Simulated vs Experimental Fractional Coverage for PC with Varying Delay Time Setting where High = 1.6sec, Base = 2.4sec, and High = 4.0sec.

4.2.3 Effect of Pre-charge Gas Pressure

The effect of varying pre-charge gas pressure setting at various rowlengths was also examined. Figure 17 shows the viscosity vs shear rate for each simulated point. It is observed that all points lie in the Newtonian region of the curve. Figure 18 shows a zoomed in version of the viscosity vs shear rate plot.

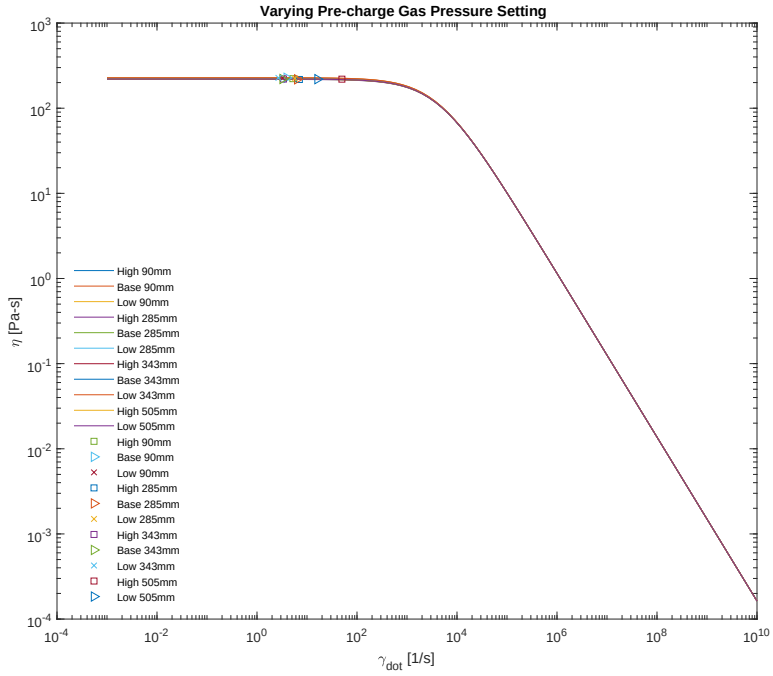


Figure 17: Viscosity vs Shear Rate for Each Simulated Point Varying Pre-charge Gas Pressure Setting for PC where High = 500psi, Base = 325psi, and Low = 175psi.

Figure 19 shows the simulated vs experimental fractional coverage as pre-charge gas pressure setting and rowlength are changed. It is observed that for the low setting, the simulated prediction tends to be slightly better due to the higher experimental fractional coverage values. Again, rowlengths 90mm, 285mm, and 343mm tend to be clustered around $m_{sim} \approx 0.9$ while at 505mm, the simulated fractional coverages tend to be closer to the experimental values.

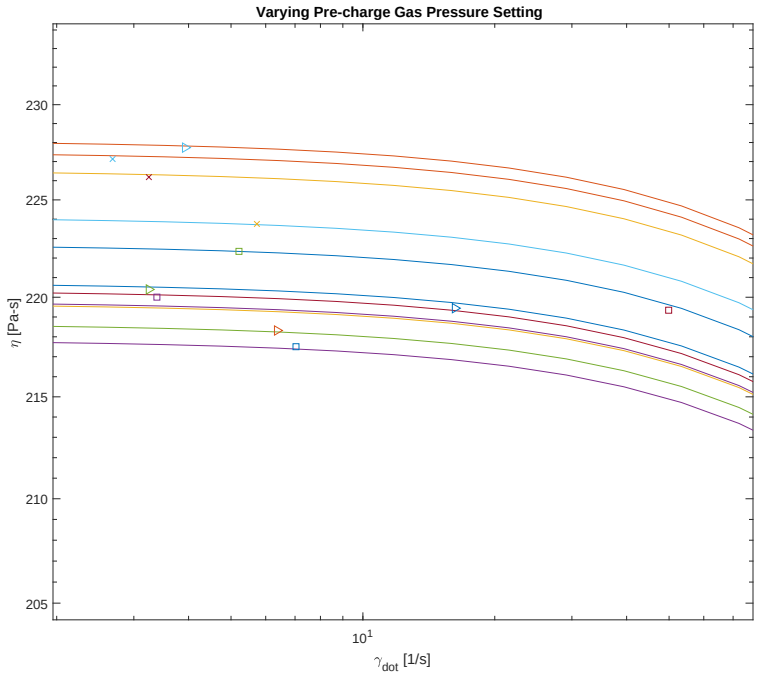


Figure 18: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Pre-charge Gas Pressure Setting for PC where High = 500psi, Base = 325psi, and Low = 175psi.

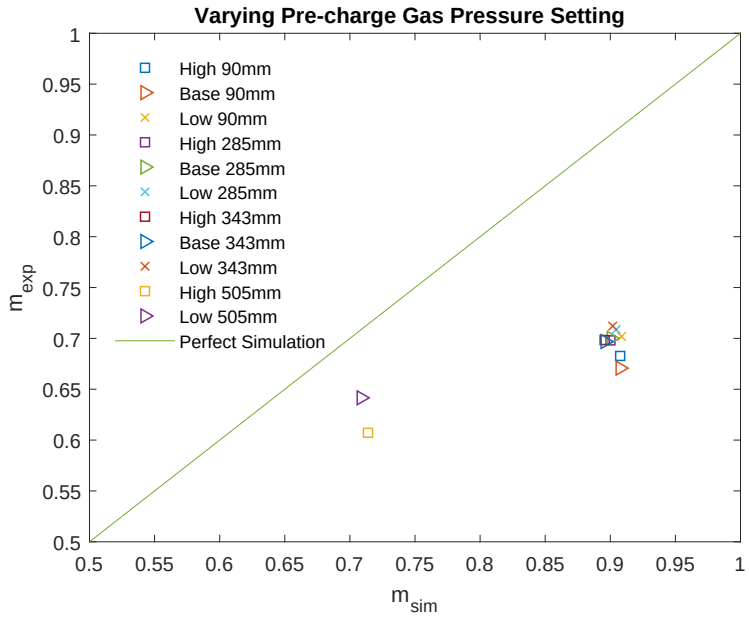


Figure 19: Simulated vs Experimental Fractional Coverage for PC with Varying Pre-charge Gas Pressure Setting where High = 500psi, Base = 325psi, and Low = 175psi.

4.2.4 Effect of Piston Speed

Lastly, the effect of varying piston speed at different rowlengths was observed. Figure 20 shows the viscosity vs shear rate for each simulated point. All points seem to lie in the Newtonian region. Figure 21 shows a zoomed in version of the viscosity vs shear rate plot to easily determine where the points lie on the curve.

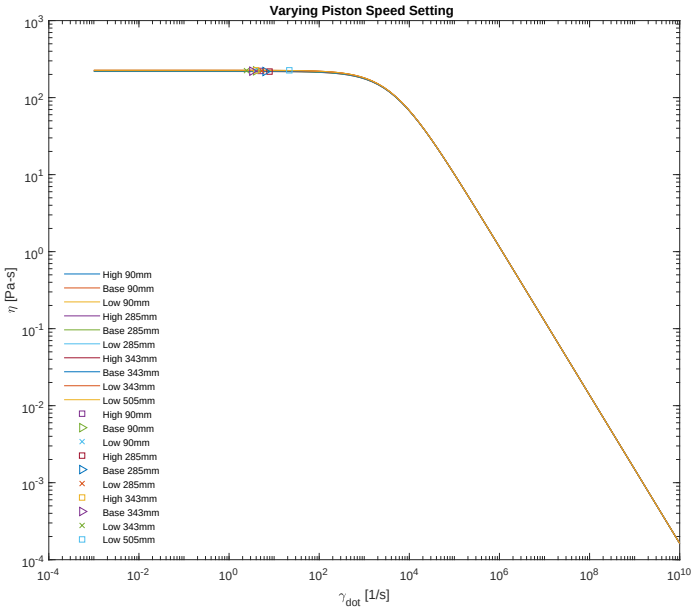


Figure 20: Viscosity vs Shear Rate for Each Simulated Point Varying Piston Speed Setting for PC where High $= 10.0\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 1.8\frac{in}{sec}$.

Figure 22 shows the simulated vs experimental fractional coverages for each scenario. For rowlengths 90mm, 285mm, and 343mm, the points tend to be clustered around $m_{sim} \approx 0.9$. No obvious trends between the points at these rowlengths are noted. However, at 505mm, the simulation seems to predict the fractional coverage more accurately.

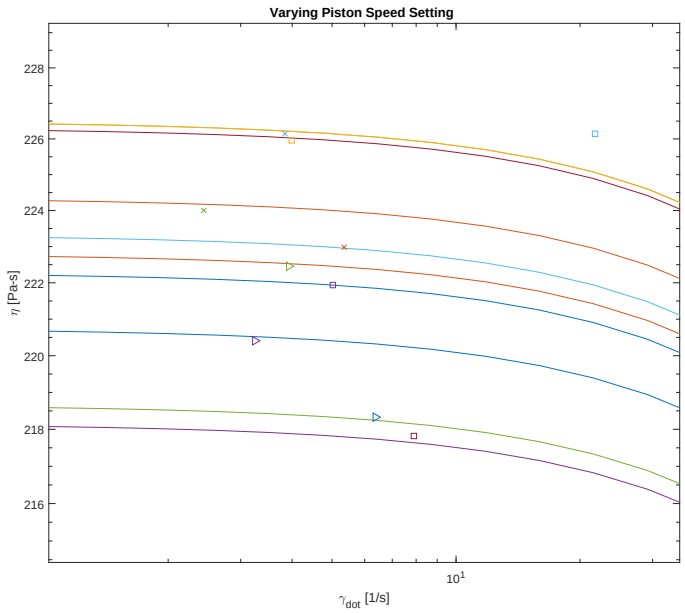


Figure 21: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Piston Speed Setting for PC where High $= 10.0\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 1.8\frac{in}{sec}$.

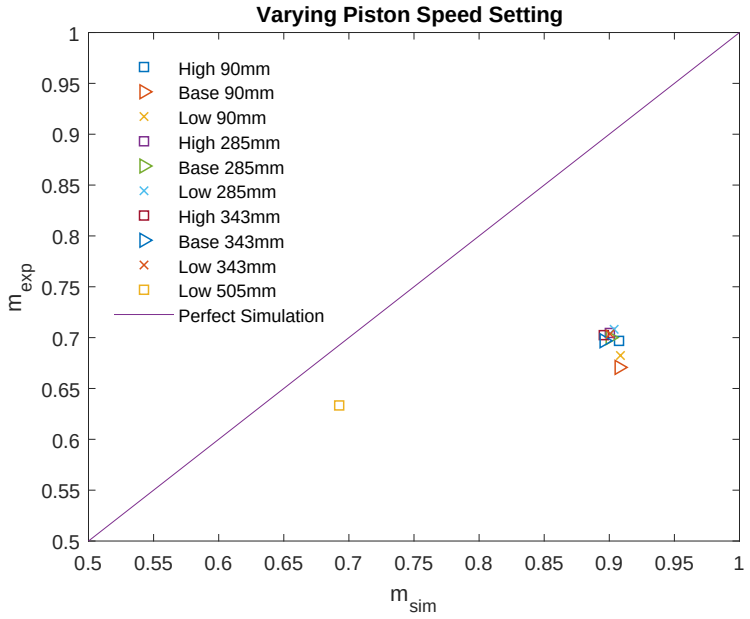


Figure 22: Simulated vs Experimental Fractional Coverage for PC with Varying Piston Speed Setting where High $= 10.0\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 1.8\frac{in}{sec}$.

4.3 Simulation Results for Non-Newtonian Fluids under Non-Isothermal Conditions

4.3.1 Temperature Profiles

The temperature profile was calculated for polystyrene in the spiral mold cavity and can be seen in Figure 23. Though three viscosity models were used to simulate the behavior of this material, only one temperature profile is shown since the viscosity model had no influence on the temperature profile. The calculated temperature profiles show the non-isothermal condition of the system as time goes on as well as the cooling of the fluid in the mold cavity.

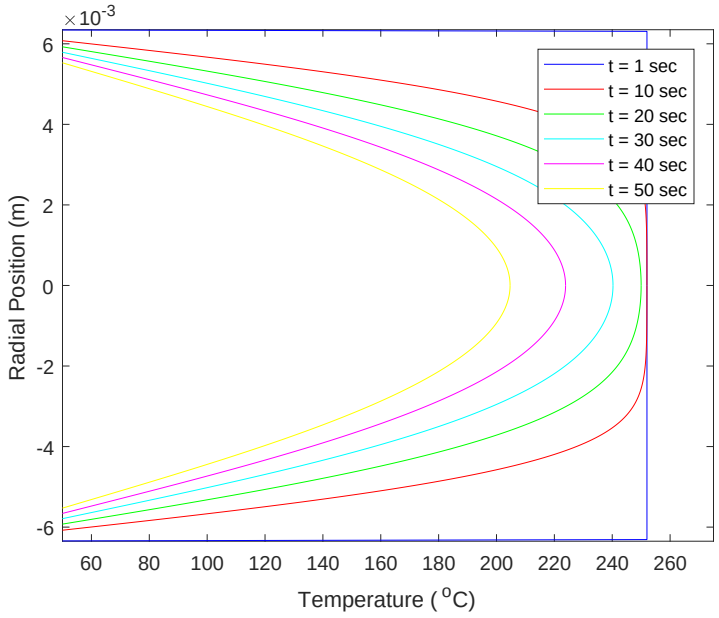


Figure 23: Temperature Profile for the Spiral Mold Cavity for PS.

4.3.2 Velocity Profiles

Normalized velocity profiles were also calculated for polystyrene for the cross-exponential, Ellis, and power law models. Figure 24 shows the normalized velocity profile calculated for polystyrene using the cross-exponential model. Again, non-isothermal velocity profiles are observed. The normalized velocity profiles show an increase in acceleration of the bubble as time increases as expected.

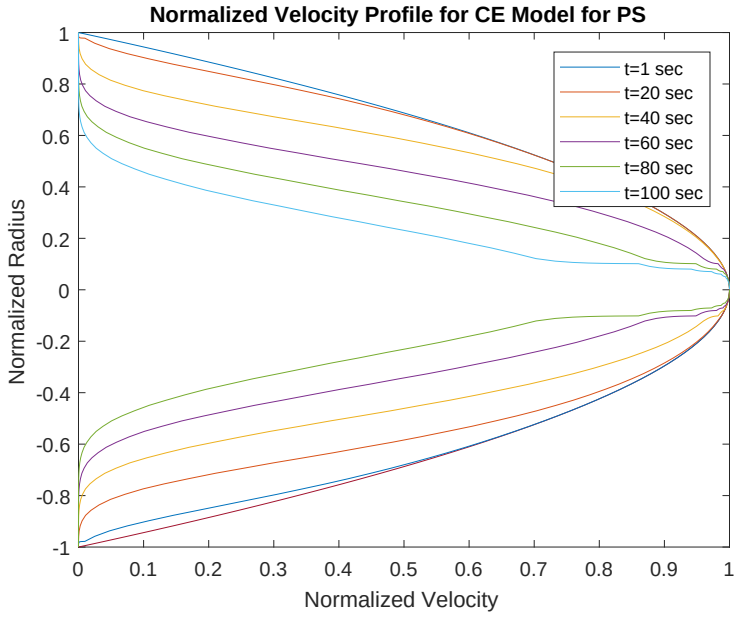


Figure 24: Normalized Velocity Profile for PS with Cross-Exponential Viscosity Model.

Figure 25 shows calculated normalized velocity profiles for polystyrene at $\tau_{wall} = 3.42$ $Pa - s^2$. Again, non-isothermal behavior and an increase in acceleration are observed.

Lastly, Figure 26 shows the calculated normalized velocity profile for polystyrene using the Power Law model. Non-isothermal behavior and increasing acceleration are observed.

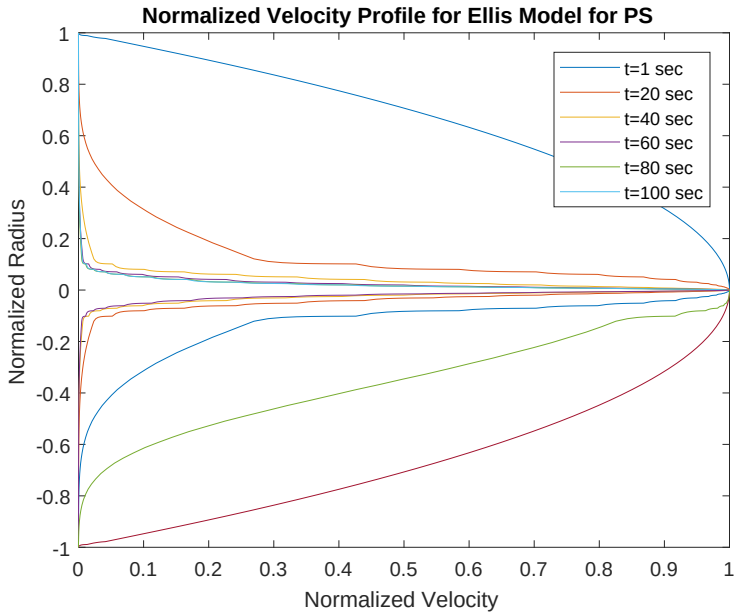


Figure 25: Normalized Velocity Profile for PS with Ellis Viscosity Model at $\tau_{wall} = 3.42Pa - s^2$.

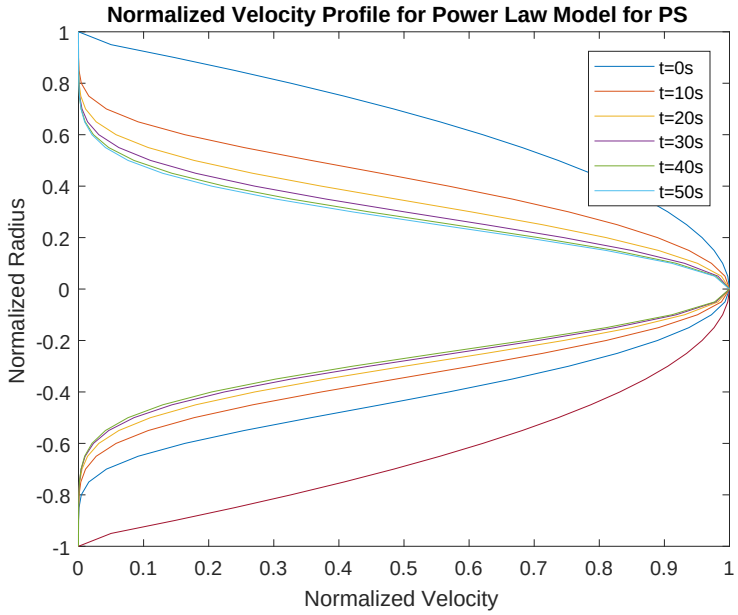


Figure 26: Normalized Velocity Profile for PS with Power Law Model.

4.3.3 m vs Fo at Base Conditions

Similar to polycarbonate, the simulated fractional coverage at base conditions was plotted against all experimental data to obtain an estimate of the simulated vs experimental results. m vs Fo plots were obtained for both the cross-exponential and Ellis models. The analysis was begun primarily by comparing the cross-exponential and Ellis models. The incorporation of the power law model was considered after.

Figure 27 shows m vs Fo for polystyrene using the cross-exponential model to describe the viscosity of the fluid. Figure 28 shows m vs Fo for polystyrene using the Ellis model to describe viscosity. Both plots tend to describe the same behavior at low Fo numbers but the Ellis model describes the elbow region of the data much more accurately than the cross-exponential model. Therefore, the Ellis model will be used for the comparison of simulated vs experimental fractional coverages.

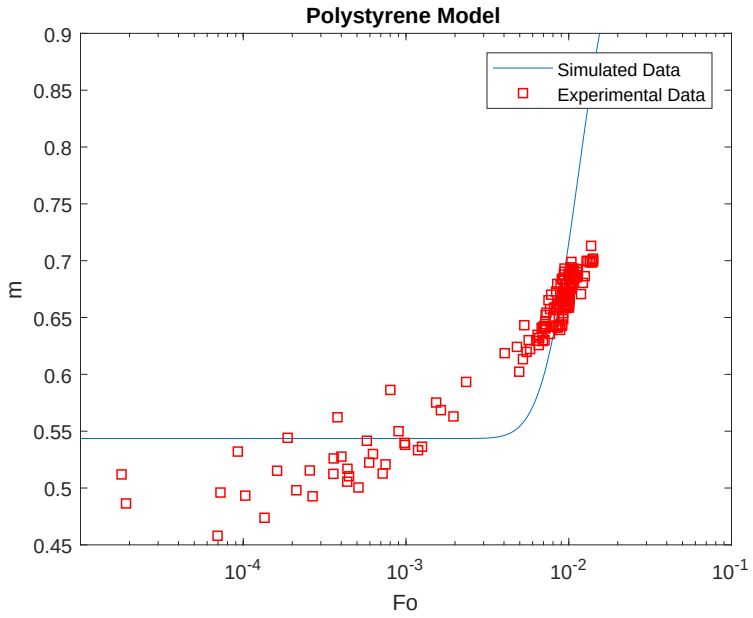


Figure 27: m vs Fo for PS with Cross-Exponential Viscosity Model at Base Processing Conditions.

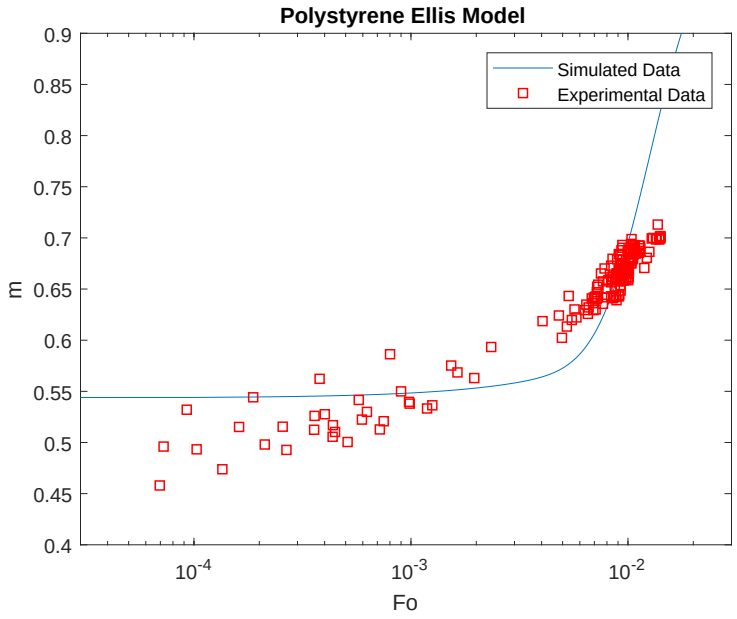


Figure 28: m vs Fo for PS with Ellis Viscosity Model at Base Processing Conditions.

4.4 Comparison of Simulation and Experimental Results for Fractional Coverage of non-Newtonian Fluids: Ellis Model

4.4.1 Effect of Shot Size

The first varied setting examined for polystyrene was shot size. Again, similar to the analysis of polycarbonate, viscosity and shear rate were determined and plotted. Figure 29 shows the viscosity vs shear rate plots for each scenario. All simulations lie in the shear-thinning region of the line, confirming our assumption that the polymer melt behaves as a non-Newtonian fluid. Figure 30 shows a zoomed in version of the viscosity vs shear rate plot for clarity.

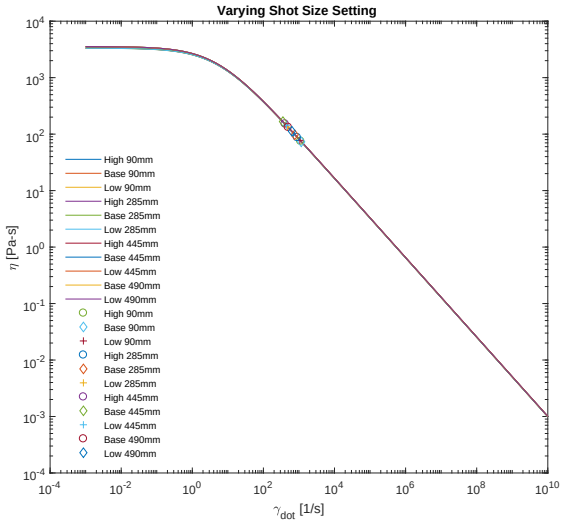


Figure 29: Viscosity vs Shear Rate for Each Simulated Point Varying Shot Size Setting Using Ellis Model for PS where High = 3.15in, Base = 3.0in, and Low = 2.85in.

Figure 31 shows the simulated vs experimental fractional coverages for polystyrene with varying shot size setting. For scenarios at low to middle rowlengths (90mm, 285mm), the simulation extremely overpredicts the fractional coverage. As rowlength increases, the simulation tends to predict a more accurate fractional coverage.

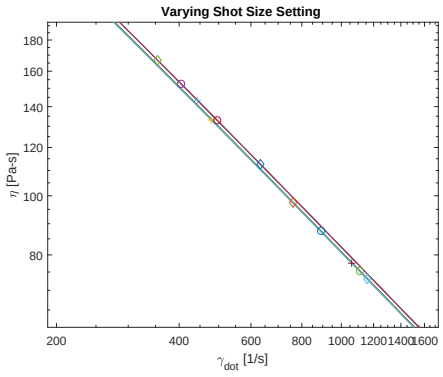


Figure 30: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Shot Size Setting Using Ellis Model for PS where High = 3.15in, Base = 3.0in, and Low = 2.85in.

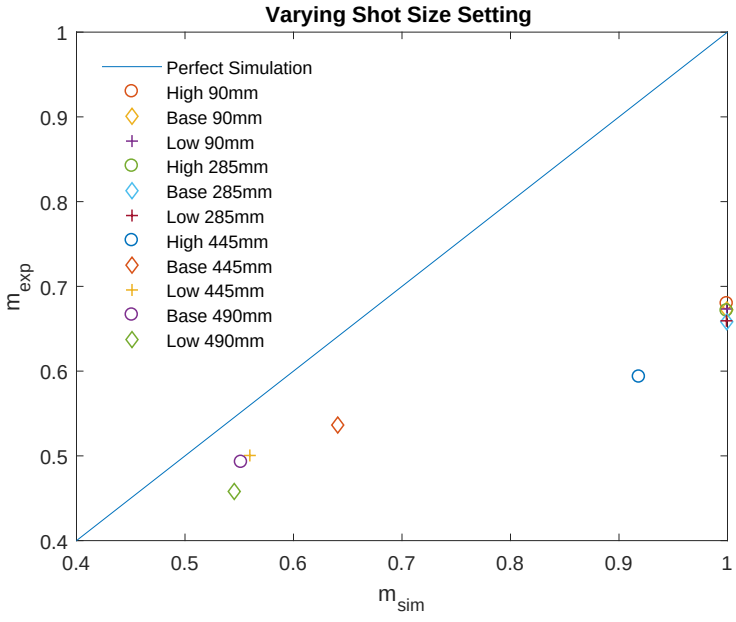


Figure 31: Simulated vs Experimental Fractional Coverage for PS with Varying Shot Size Setting Using Ellis Model where High = 3.15in, Base = 3.0in, and Low = 2.85in.

4.4.2 Effect of Delay Time

The next observed effect was the variation of delay time setting. Again, viscosity vs shear rate was plotted. Figure 32 shows the viscosity vs shear rate for polystyrene with varying delay time setting. We observe that all examined points lie in the shear-thinning portion of the curve suggesting the polystyrene behaved as a non-Newtonian fluid. Figure 33 shows a zoomed in version of the viscosity vs shear rate plot.

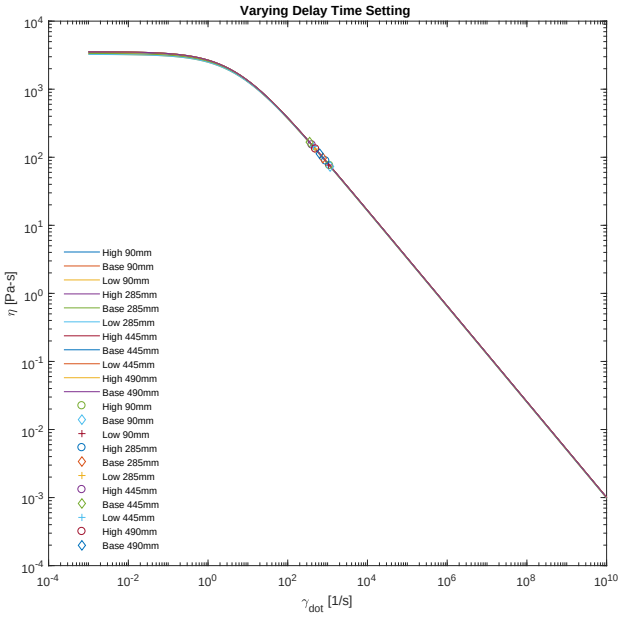


Figure 32: Viscosity vs Shear Rate for Each Simulated Point Varying Delay Time Setting Using Ellis Model for PS where High = 4.37sec, Base = 2.37sec, and Low = 0.62sec.

Figure 34 shows the simulated vs experimental fractional coverages for each tested condition. Similar to the shot size setting, the simulation highly overpredicts the fractional coverage at 90mm and 285mm but provides more accurate predictions at 445mm and 490mm.

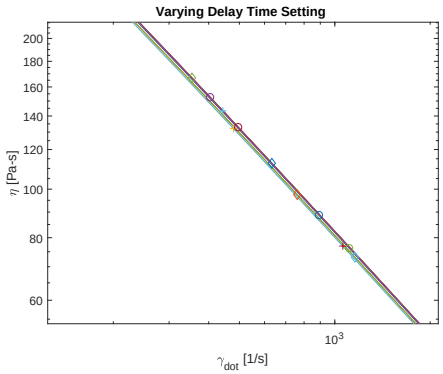


Figure 33: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Delay Time Setting Using Ellis Model for PS where High = 4.37sec, Base = 2.37sec, and Low = 0.62sec.

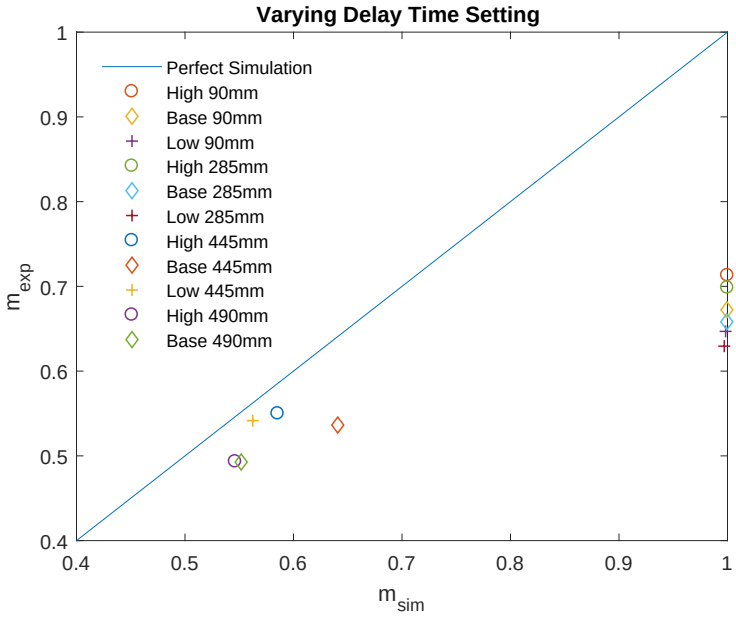


Figure 34: Simulated vs Experimental Fractional Coverage for PS with Varying Delay Time Setting Using Ellis Model where High = 4.37sec, Base = 2.37sec, and Low = 0.62sec.

4.4.3 Effect of Pre-charge Gas Pressure

The pre-charge gas pressure setting was then varied. Non-Newtonian behavior was noticed with the viscosity vs shear rate plot in Figure 35 for each data point. Figure 36 shows a zoomed-in version of the viscosity vs shear rate plot to observe the subtle differences in temperature between each scenario.

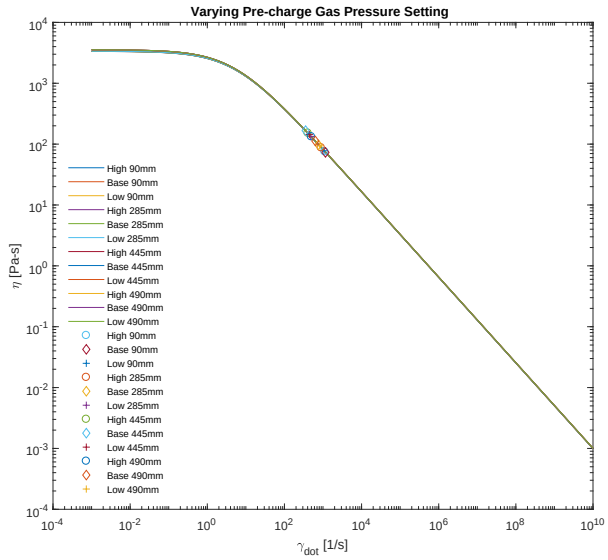


Figure 35: Viscosity vs Shear Rate for Each Simulated Point Varying Pre-charge Gas Pressure Setting Using Ellis Model for PS where High = 500psi, Base = 350psi, and Low = 200psi.

Figure 37 shows the experimental vs simulated fractional coverages for polystyrene with varying pre-charge gas pressure setting. Again, at rowlengths 90mm and 285mm the simulation overpredicts the fractional coverage. At rowlengths 445mm and 490mm the predictions become more accurate. We can also observe that for the low pre-charge gas pressure setting at 490mm the simulation prediction is almost the same (slightly lower) as the experimental value.

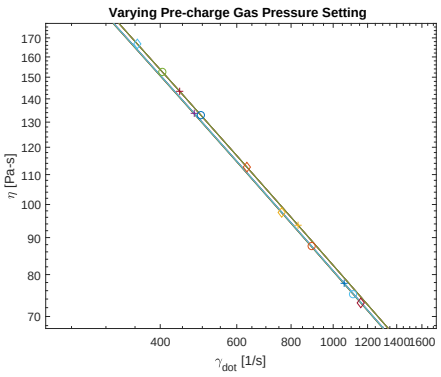


Figure 36: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Pre-charge Gas Pressure Setting Using Ellis Model for PS where High = 500psi, Base = 350psi, and Low = 200psi.

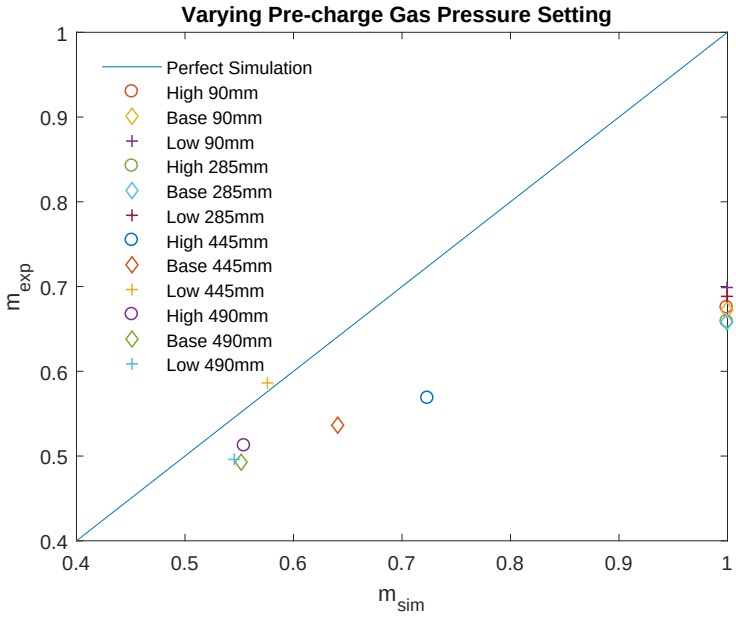


Figure 37: Simulated vs Experimental Fractional Coverage for PS with Varying Pre-charge Gas Pressure Setting Using Ellis Model where High = 500psi, Base = 350psi, and Low = 200psi.

4.4.4 Effect of Piston Speed

The piston speed setting was then varied. Figure 38 confirms the assumption that polystyrene behaved as a non-Newtonian fluid. Figure 39 shows a zoomed-in version of the viscosity vs shear rate plot.

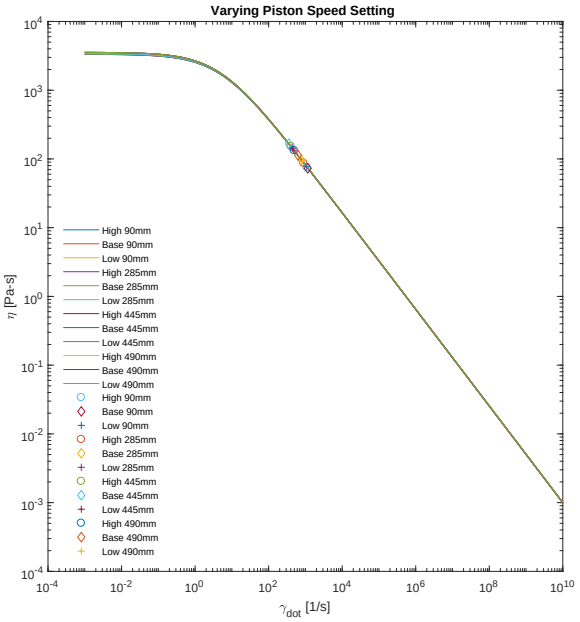


Figure 38: Viscosity vs Shear Rate for Each Simulated Point Varying Piston Speed Setting Using Ellis Model for PS where High $= 10.0\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 1.8\frac{in}{sec}$.

Figure 40 shows the simulated vs experimental fractional coverages for polystyrene with varying piston speed setting at several rowlengths. The simulation does not predict well the fractional coverage at rowlengths 90mm and 285mm for all piston speed settings. Rowlength 490mm simulated the most accurate prediction for all piston speed settings.

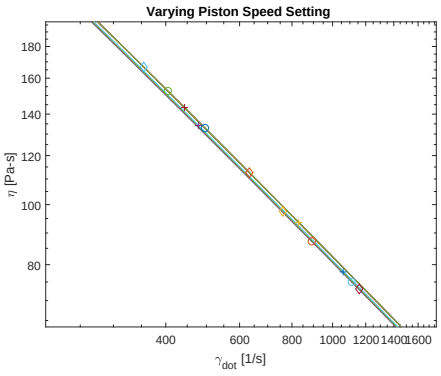


Figure 39: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Piston Speed Setting Using Ellis Model for PS where High = $10.0\frac{in}{sec}$, Base = $3.1\frac{in}{sec}$, and Low = $1.8\frac{in}{sec}$.

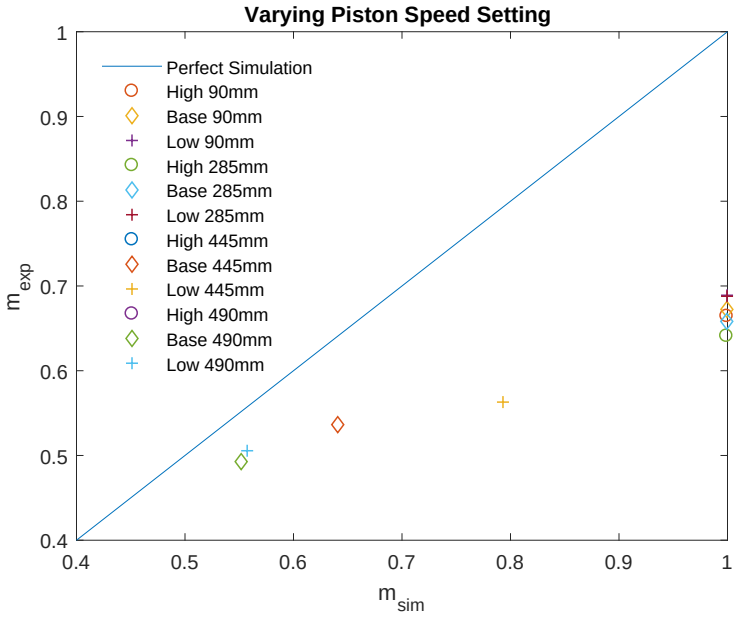


Figure 40: Simulated vs Experimental Fractional Coverage for PS with Varying Piston Speed Setting Using Ellis Model where High = $10.0\frac{in}{sec}$, Base = $3.1\frac{in}{sec}$, and Low = $1.8\frac{in}{sec}$.

4.4.5 Effect of Melt Temperature

The last varied setting examined was melt temperature. Figure 41 shows the viscosity vs shear rate plot for each scenario. It is interesting to observe that due to the variation of melt temperature, the viscosity vs shear rate lines shift into three different clustered lines due to the three settings of melt temperature tested. Again we observe the polystyrene behaved as a non-Newtonian fluid for all scenarios. Figure 42 shows a zoomed-in version of Figure 43.

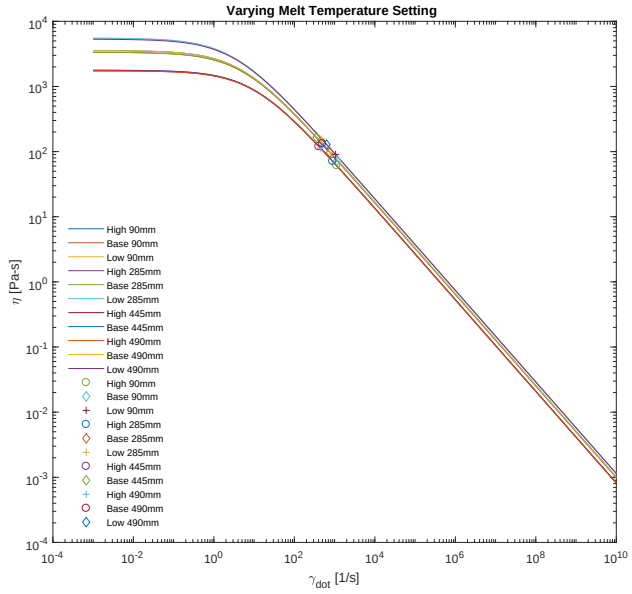


Figure 41: Viscosity vs Shear Rate for Each Simulated Point Varying Melt Temperature Setting Using Ellis Model for PS where High = 511.7^{o}F, Base = 485.6^{o}F, and Low = 471.5^{o}F.

Figure 43 shows the simulated vs experimental fractional coverages for polystyrene at different rowlengths and melt temperature settings. The simulation does not predict the fractional coverage at 90mm and 285mm well but predicts the fractional coverages at 490mm almost exactly for all melt temperature settings. The simulated fractional coverages at 445mm were much more accurately predicted compared to 90mm and 285mm but are not exact.

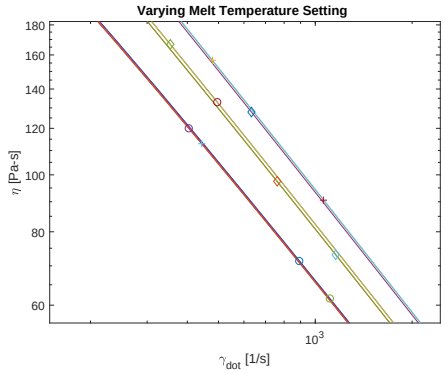


Figure 42: Zoomed Viscosity vs Shear Rate for Each Simulated Point Varying Melt Temperature Setting Using Ellis Model for PS where High = 511.7^{o}F, Base = 485.6^{o}F, and Low = 471.5^{o}F.

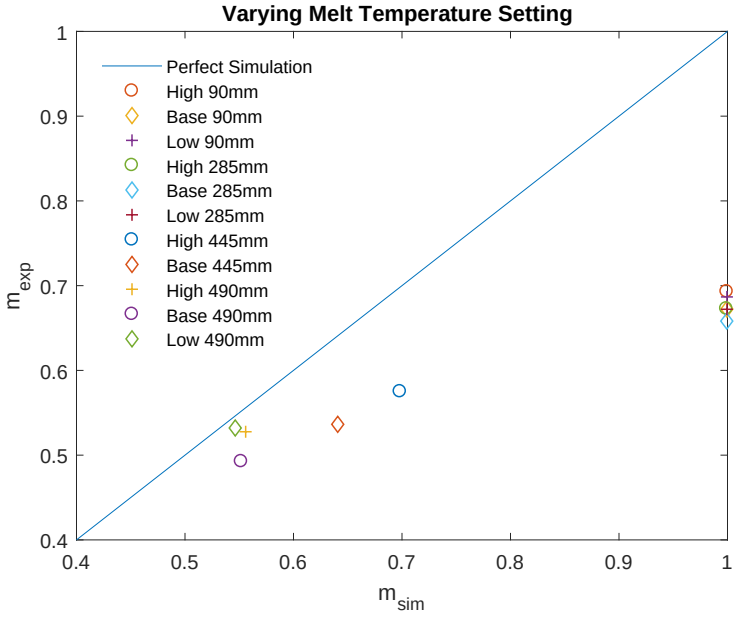


Figure 43: Simulated vs Experimental Fractional Coverage for PS with Varying Melt Temperature Setting Using Ellis Model where High = 511.7^{o}F, Base = 485.6^{o}F, and Low = 471.5^{o}F.

4.5 Comparison of Simulation and Experimental Results for Fractional Coverage of non-Newtonian Fluids: Power Law Model

After obtaining the results using the Ellis model, the Power Law model was used to attempt to obtain more accurate simulation responses at lower rowlengths. In this section, viscosity vs shear rate was not plotted due to the power law behavior noted in the previous sections of this work. Rowlengths of 90mm, 285mm, 445mm, and 490mm were chosen, the same as those used for the Ellis model.

4.5.1 Effect of Shot Size

The first varied setting considered was shot size. Figure 44 shows the obtained results using the power law model for varying shot size setting. All simulated results are higher than the experimental results. At rowlengths of 90mm and 285mm, m_{sim} lies around 0.9 for all simulations, at least 0.2 higher than any of the experimental results. A higher fractional coverage prediction would infer a higher bubble radius prediction. This could indicate that the simulated heat transfer is occurring faster compared to reality. Similar trends appear with 445mm and 490mm rowlengths. m_{sim} is at least 0.1 higher than the experimental results.

4.5.2 Effect of Delay Time

The effect of delay time was then observed. Figure 45 shows the obtained results using the power law model while varying the delay time setting. Again m_{sim} tends to be at least 0.1 higher than the experimental results. It is also interesting to note for this setting that the high, base, and low settings at 90mm and 285mm are predicted to be almost exactly the same values despite their differences in rowlengths.

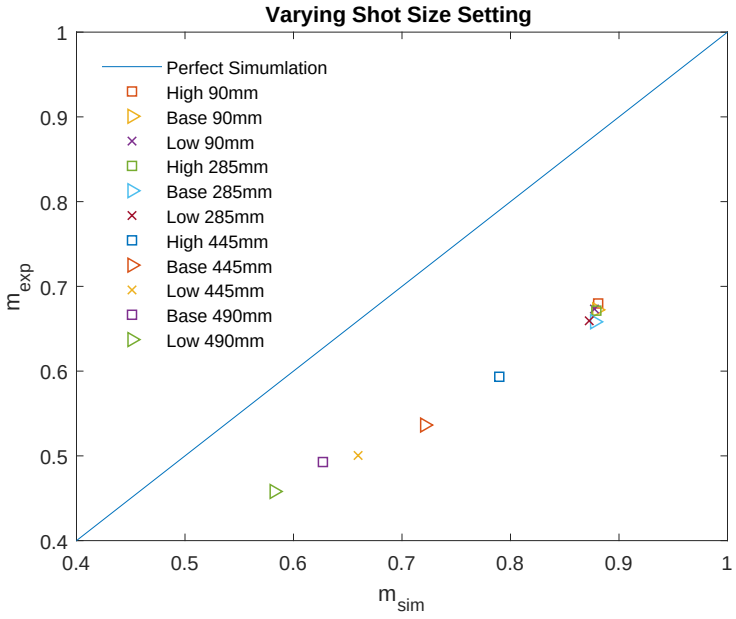


Figure 44: Simulated vs Experimental Fractional Coverage for PS Using the Power Law Model with Varying Shot Size Setting where High = 3.15in, Base = 3.0in, and Low = 2.85in.

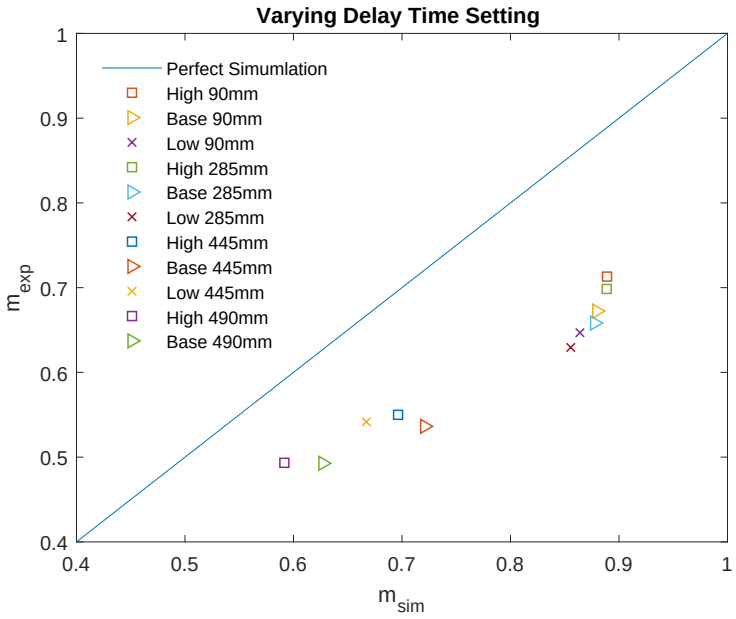


Figure 45: Simulated vs Experimental Fractional Coverage for PS Using the Power Law Model with Varying Delay Time Setting where High = 4.37sec, Base = 2.37sec, and Low = 0.62sec.

4.5.3 Effect of Pre-charge Gas Pressure

Figure 46 shows the obtained results using the power law model while varying the pres-charge gas pressure setting. When observing the varying effects of this setting, m_{sim} is exactly the same at 90mm and 285mm for all settings. This is not true for either 445mm or 490mm. Additionally, the predictions of this model tend to be 0.1-0.3 greater than the experimental fractional coverages.

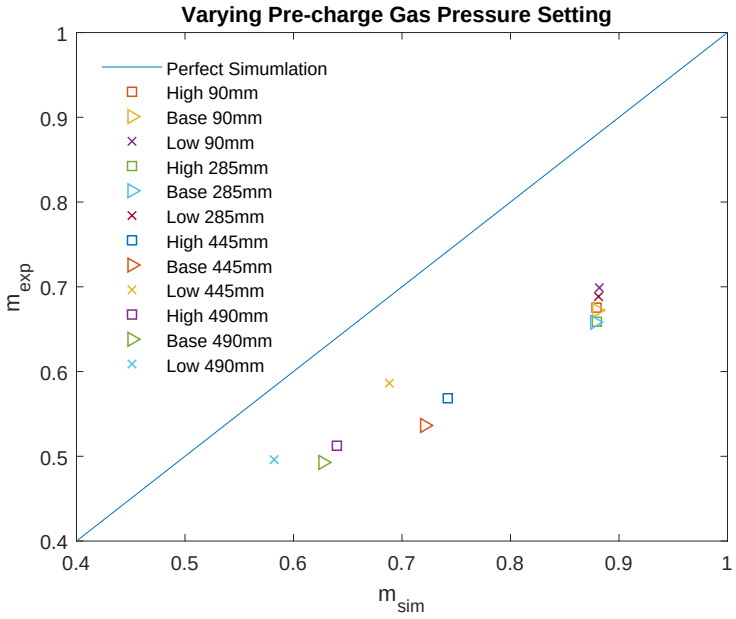


Figure 46: Simulated vs Experimental Fractional Coverage for PS Using the Power Law Model with Varying Pre-charge Gas Pressure Setting where High = 500psi, Base = 350psi, and Low = 200psi.

4.5.4 Effect of Piston Speed

Next, the effect of varying piston speed setting was observed. Figure 47 shows the simulated vs experimental fractional coverages while varying piston speed setting. It should be noted than no experimental results were obtained for the High setting at 445mm or 490mm rowlengths. Again, $m_{sim} \approx 0.9$ for all settings at 90mm and 285mm. This fact is not true at 445mm and 490mm but m_{sim} is still overpredicted for all rowlengths and all settings.

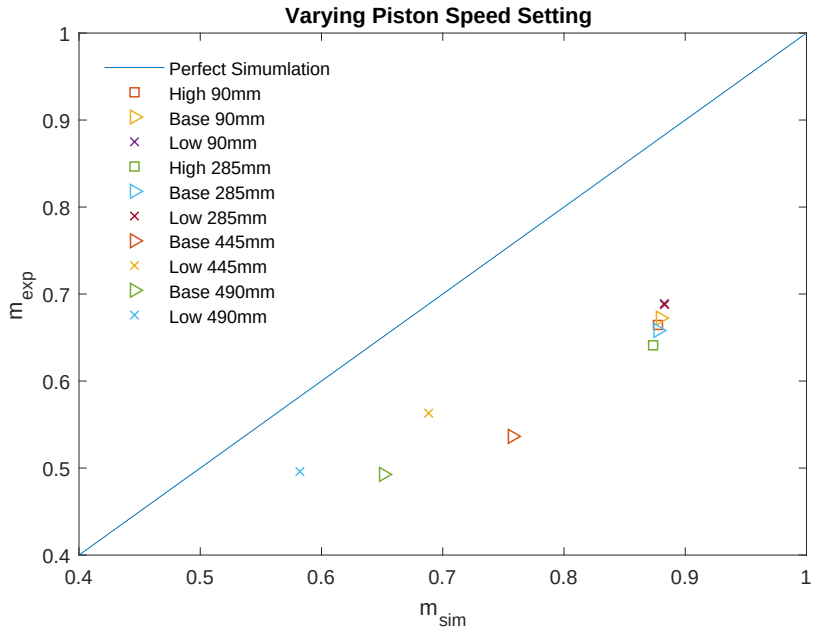


Figure 47: Simulated vs Experimental Fractional Coverage for PS Using the Power Law Model with Varying Piston Speed Setting where High $= 10\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 2.8\frac{in}{sec}$.

4.5.5 Effect of Melt Temperature

Lastly, the effect of melt temperature was observed. Figure 48 shows the simulated vs experimental fractional coverages for polystyrene with varying melt temperature setting. Simulated fractional coverage overpredicts fractional coverage for all settings and rowlengths tested. Similar to other varied variables, simulated fractional coverage for 90mm and 285mm rowlengths are exactly the same. While this does not hold true for 445mm and 490mm, m_{sim} is still 0.1-0.3 higher than m_{exp}.

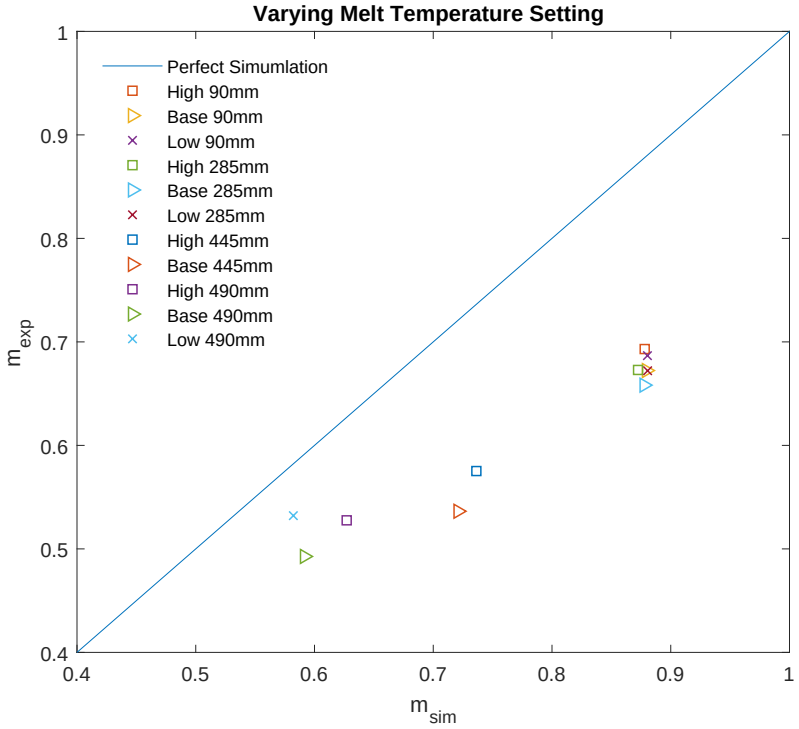


Figure 48: Simulated vs Experimental Fractional Coverage for PS Using the Power Law Model with Varying Melt Temperature Setting where High = 511.7^{o}F, Base = 485.6^{o}F, and Low = 471.5^{o}F.

4.6 Comparison of Simulation Results for non-Newtonian Fluids under Non-Isothermal Conditions

The simulated results from both the Ellis and Power Law models were then plotted jointly to compare the predictions of the two models for each varied setting and rowlength tested.

4.6.1 Effect of Shot Size

Figure 49 shows the simulated fractional coverages using both the Ellis and power law models against experimental fractional coverage. Overall, the Ellis model predicts fractional coverage more accurately at 445mm and 490mm while the power law model predicts fractional coverage more accurately at 90mm and 285mm. This is not to say that their predictions are precise but rather to highlight the advantages of each model.

4.6.2 Effect of Delay Time

Figure 50 shows the simulated fractional coverages using both the Ellis and power law models against experimental fractional coverage. Again, the simulation utilizing the Ellis model predicts fractional coverage at 445mm and 490mm more accurately than the simulation utilizing the power law model while the power law model simulation predicts fractional coverage more accurately an 90mm and 285mm than the Ellis model simulation.

4.6.3 Effect of Pre-charge Gas Pressure

Figure 51 plots simulated fractional coverage vs experimental fractional coverage for both the Ellis model and power law model simulations. The Ellis model simulation more accurately predicts the fractional coverage at the higher rowlengths while the power law model more accurately predicts the fractional coverage at lower rowlengths.

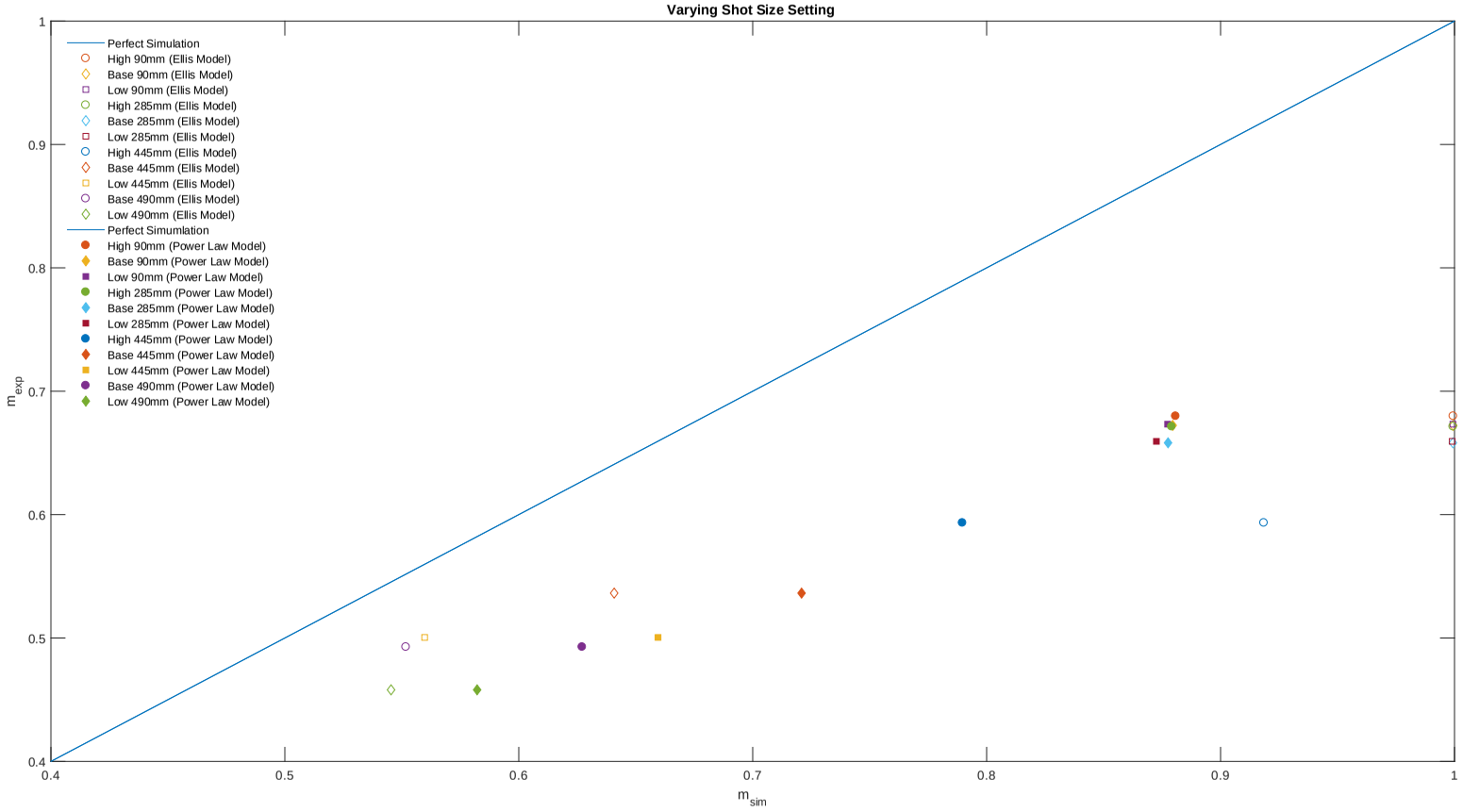


Figure 49: Comparison of Simulated vs Experimental Fractional Coverage for PS of Both the Ellis and Power Law Models with Varying Shot Size Setting where High = 3.15in, Base = 3.0in, and Low = 2.85in.

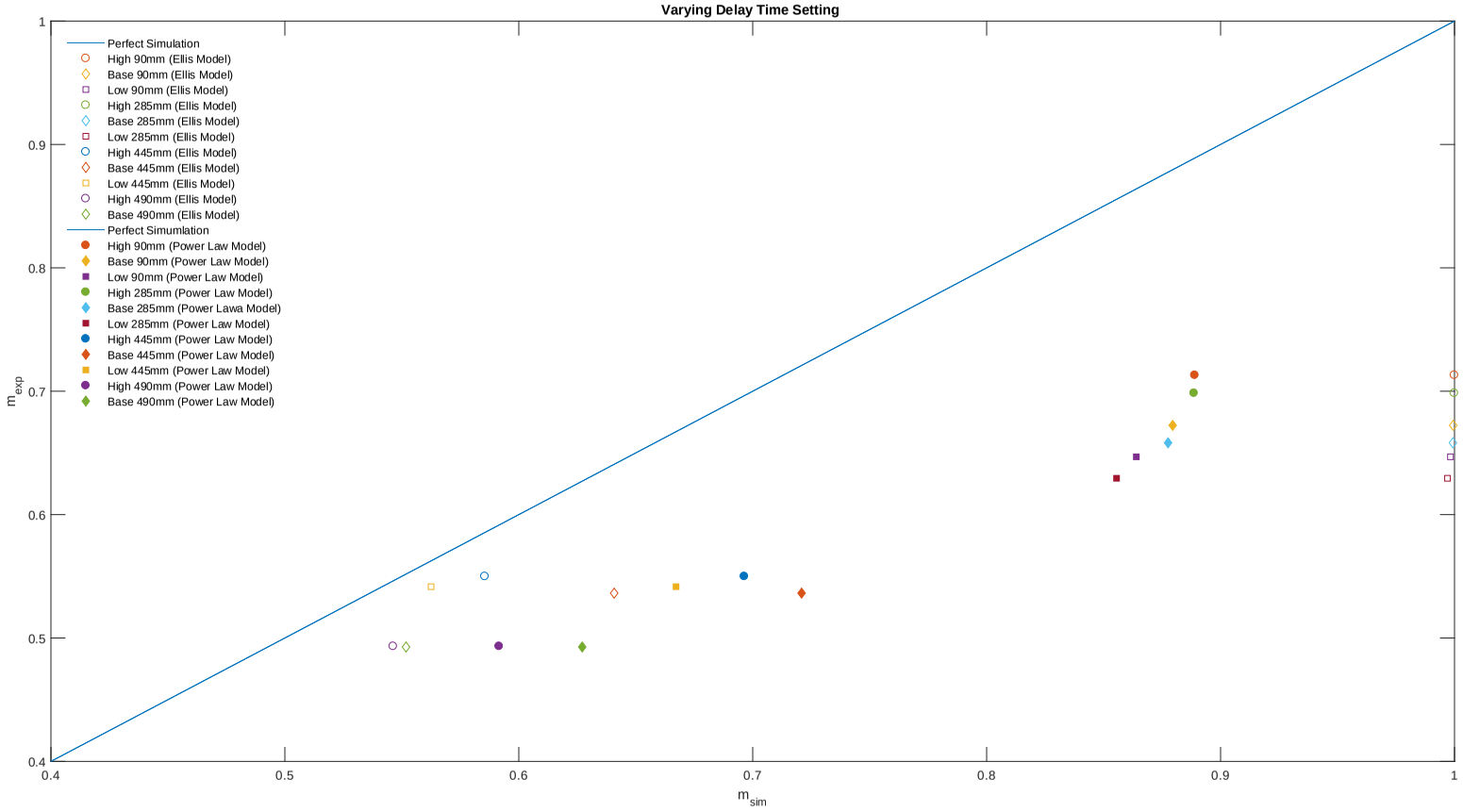


Figure 50: Comparison of Simulated vs Experimental Fractional Coverage for PS of Both the Ellis and Power Law Models with Varying Delay Time Setting where High = 4.37sec, Base = 2.37sec, and Low = 0.62sec.

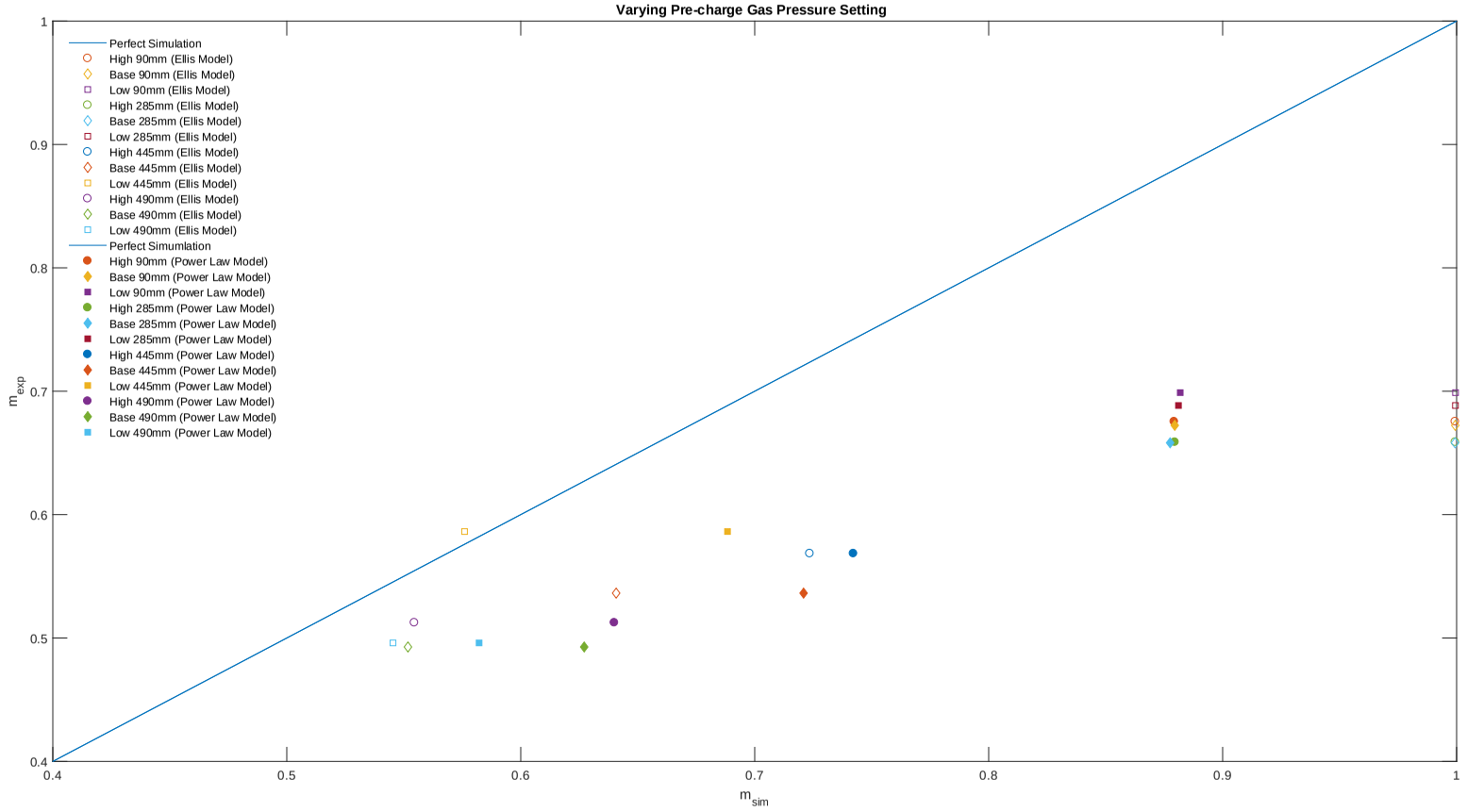


Figure 51: Comparison of Simulated vs Experimental Fractional Coverage for PS of Both the Ellis and Power Law Models with Varying Pre-charge Gas Pressure Setting where High = 500psi, Base = 350psi, and Low = 200psi.

4.6.4 Effect of Piston Speed

Figure 52 shows the simulated fractional coverages using both the Ellis and power law models against experimental fractional coverage. Once again, the Ellis model simulation is more accurate at 445mm and 490mm while the power law model is more accurate at 90mm and 285mm.

4.6.5 Effect of Melt Temperature

Figure 53 plots the experimental fractional coverage vs the simulated fractional coverages calculated by using either the power law model or the Ellis model. The power law model more accurately predicts fractional coverage at 90mm and 285mm compared to the Ellis model while the Ellis model more accurately predicts fractional coverage at 445mm and 490mm compared to the power law model simulations. The predicted fractional coverages are almost always higher than the experimental values which could indicate a flaw in the simulation model, whether it be an incorrect assumption or relation.

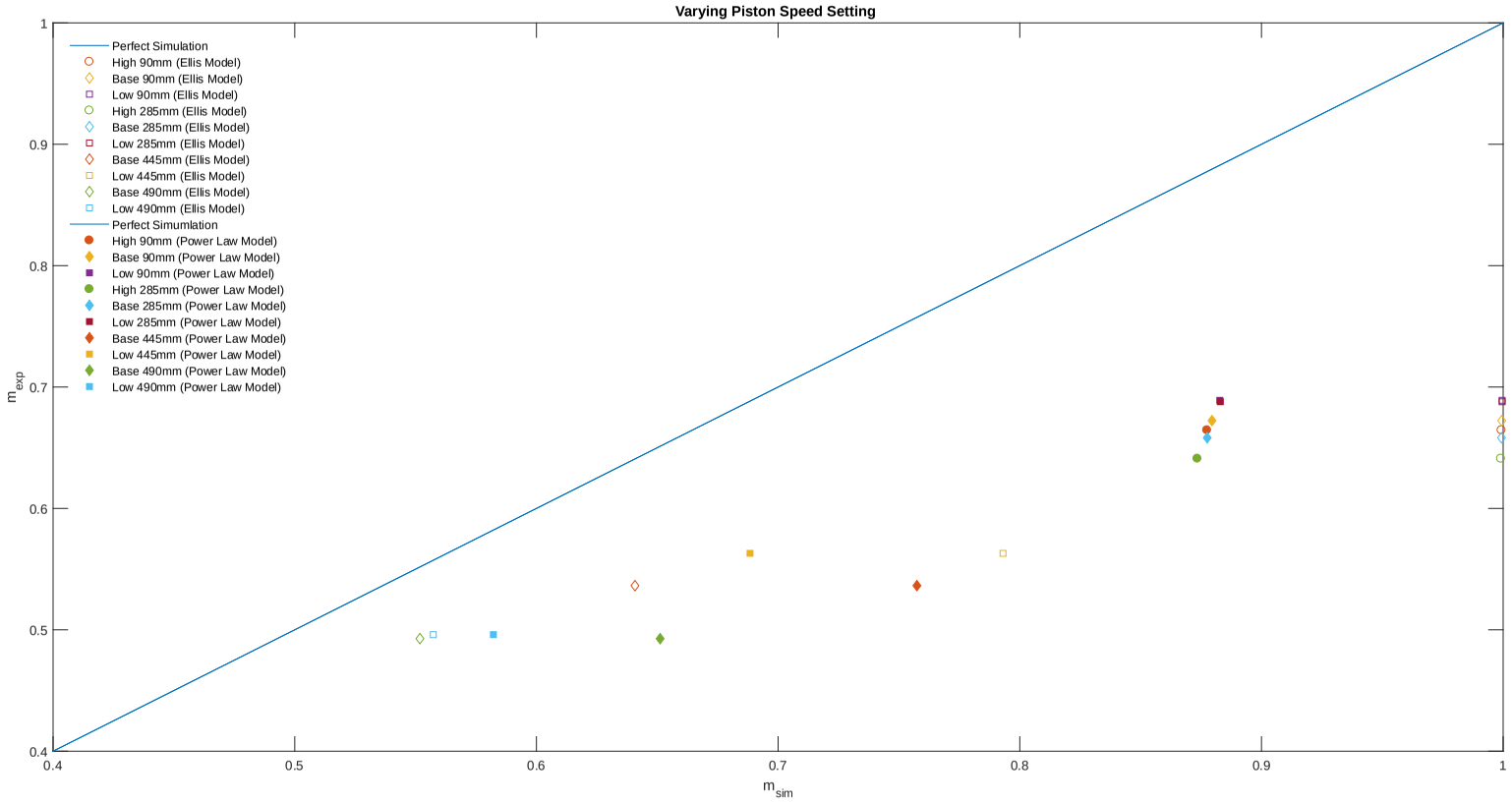


Figure 52: Comparison of Simulated vs Experimental Fractional Coverage for PS of Both the Ellis and Power Law Models with Varying Shot Size Setting where High $= 10\frac{in}{sec}$, Base $= 3.1\frac{in}{sec}$, and Low $= 2.8\frac{in}{sec}$.

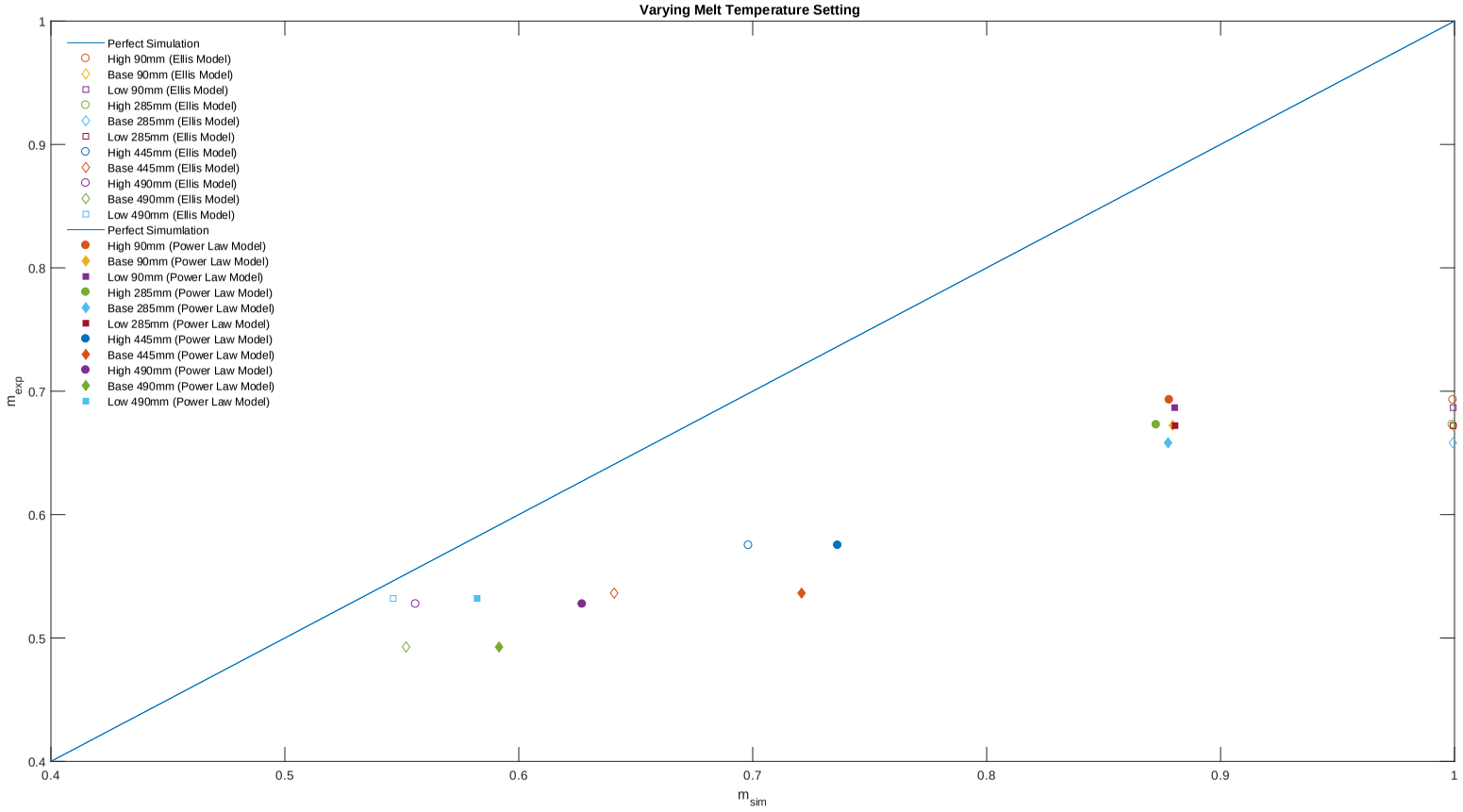


Figure 53: Comparison of Simulated vs Experimental Fractional Coverage for PS of Both the Ellis and Power Law Models with Varying Melt Temperature Setting where High = 511.7^{o}F, Base = 485.6^{o}F, and Low = 471.5^{o}F.

5 Conclusion

Three major simulation models for predicting fractional coverage using gas-assisted injection of polymer melts in tube geometries were presented. The first model presented was for fluids following Newtonian behavior. The cross-exponential zero-shear viscosity model was used and the determined fractional coverage values were compared with experimental data collected by Kaminski [9]. Viscosity vs shear rate plots were obtained for all tested scenarios and showed that the assumption that polycarbonate behaved as a Newtonian fluid was valid. The comparison of the simulated and experimental data showed good predictions at further rowlengths than those closer to the entrance. In addition, the variation of several process parameters were examined. No major trends were found between the variation of settings for each parameter. Further, velocity profiles and temperature profiles were calculated and plotted and the non-isothermal behavior was shown to be captured by the system as well as the cooling of the tube and narrowing bubble as time passed.

The non-Newtonian modeling began by the determination of temperature and velocity profiles using both the Ellis and cross-exponential viscosity models. Both models showed non-isothermal behavior as time passed as well as fluid cooling and the narrowing of the bubble through the tube. It was observed that at the base conditions for all data points, the Ellis model seemed to predict the fractional coverage more accurately than the cross-exponential model. Therefore, the Ellis model was used to determine simulated fractional coverages for several process parameters settings and rowlengths. Viscosity vs shear rate was plotted for all tested scenarios and it was determined that the polystyrene behaved as a non-Newtonian fluid during these experiments. The simulated vs experimental fractional coverage plots again showed better predictions as rowlength increased. Several process parameters were varied and examined when plotting the simulated vs experimental fractional coverages. No major trends between the levels of each setting were observed.

The Ellis model extremely overpredicted fractional covereage values at low rowlengths. The power law model was incorporated to attempt to obtain more accurate predictions at lower rowlengths compared to the Ellis model. With the incorporation of the power

law model, more accurate predictions of the fractional coverage at lower rowlengths were determined albeit still higher than a perfect simulation. Overall, for the non-Newtonian modeling, the Ellis model tended to be more accurate at higher rowlengths while the power law model was more accurate at lower rowlengths.

6 Future Work

The models presented take into consideration multiple assumptions that limit the prediction models. The most limiting assumption is that of high capillary number. This is presented in the calculation of asymptotic isothermal fractional coverage. The model also assumed forced convection during the calculation of the heat transfer coefficient. This model would not be valid in cases where free convection is the dominant form of heat transfer (i.e. a tube in a water bath). The models are also limited by the viscosity models chosen to model the fluids. If chosen fluids do not follow the viscosity model chosen, the model is no longer valid. In addition, the modeling does not take into consideration the effects of viscous heating. Lastly, the temperature dependence of viscosity models were taken at face value. Sensitivity studies on these temperature dependence of these models regarding the system could be completed.

The model could be expanded to eliminate the stated limitations. This work could also be expanded to examine the effect of capillary number on the system as well as the expansion of the model to incorporate different geometries. Studies regarding the effects of viscous heating on the system could also be determined as to whether this phenomena greatly affects the system or dissipates in the loading tip of the mold.

7 References

[1] F. Belblidia, J.f.t. Pittman, A. Polynkin, and J. Sienz. Gas displacing viscous shear thinning liquids from tubes: Effects of cooling before gas injection. *Chemical Engineering Science*, 60(17):49534956, 2005.

[2] Shia-Chung Chen, Sheng-Yan Hu, Sher-Meng Chao, and Rean-Der Chien. Simulation of a mold-cooling process for gas-assisted injection molded parts designed with a top rib on the gas channel. *Polymer Engineering and Science*, 40(3):595606, 2000.

[3] S. W. Churchill. A correlating equation for forced convection from gases and liquids to a circular cylinder. *Journal of Heat Transfer*, 99(2):300, 1977.

[4] B. G. Cox. On driving a viscous fluid out of a tube. *Journal of Fluid Mechanics*, 14(01):8196, 1962.

[5] Fred Fairbrother and A. E. Stubbs. Studies in electro-endosmosis. part vi. the "bubble-tube" method of measurement. *Journal of the Chemical Society*, page 527529, 1935.

[6] Vishal Gauri and Kurt W. Koelling. The motion of long bubbles through viscoelastic fluids in capillary tubes. *Rheologica Acta*, 38(5):458470, May 1999.

[7] Michael Hansen. Gas-assist injection molding: An innovative medical technology. *Medical Device and Diagnostic Technology Online*, Aug 2005.

[8] P.c Huzyak and K.w Koelling. The penetration of a long bubble through a viscoelastic fluid in a tube. *Journal of Non-Newtonian Fluid Mechanics*, 71(1-2):7388, 1997.

[9] Ronald C. Kaminski. *Gas-Assisted Injection Molding: Influence of Processing Conditions and Material Properties*. PhD thesis, 1996.

[10] Jianhui Li, Lei Chen, Huamin Zhou, and Dequn Li. Surface model based modeling and simulation of filling process in gas-assisted injection molding. *Journal of Manufacturing Science and Engineering*, 131(1):011008, 2009.

[11] A. J. Poslinski and D. J. Coyle. Steady gas penetration through non-newtonian liquids in tube and slit geometries: isothermal shear thinning effects. *Proc. Polymer Processing Society, 10th Annual Meeting*, page 219, 1994.

[12] A. J. Poslinski, P. R. Oehler, and V. K. Stokes. Isothermal gas-assisted displacement of viscoplastic liquids in tubes. *Polymer Engineering and Science*, 35(11):877892, 1995.

[13] Y.k. Shen. The study on polymer melt front, gas front and solid layer in filling stage of gas-assisted injection molding. *International Communications in Heat and Mass Transfer*, 28(1):139148, 2001.

[14] G. I. Taylor. Deposition of a viscous fluid on the wall of a tube. *Journal of Fluid Mechanics*, 10(02):161165, 1961.

[15] Yijie Wang. *The Effect of non-Newtonian Rheology on Gas-Assisted Injection Molding Process.* PhD thesis, 2003.

8 Appendix A: Abbreviations and Notation

α: degree of shear-thinning behavior

α_p: thermal diffusivity of polymer melt

α_s: thermal diffusivity of steel

A_p: Cross-sectional area of polymer deposited on tube walls by gas bubble

A_t: Tube cross-sectional area

B: coefficient dependent on molecular weight and temperature

Ca: Capillary number

dr_2: radial step change

ΔP: pressure gradient

η: Fluid viscosity

η_o: Zero Shear Viscosity

FDM: Finite Difference Method

FEM: Finite Element Method

Fo: Fourier number

GAIM: Gas-assisted Injection Molding

$\dot{\gamma}$: Shear rate

h: convective heat transfer coefficient

k_s: thermal conductivity of stainless steel

L: tube length

m: Fractional coverage

m_∞: isothermal asymptotic fractional coverage

n: shear-thinning coefficient in cross-exponential model

n: power law index

Nu: Nusselt number

Pr: Prandtl number

PC: Polycarbonate

PS: Polystyrene

Q: volumetric flow rate

r_p: radial position of polymer

r_s: radial position of steel

Ra: Raleigh number

R_{out}: Radius of outer tube

R_t: Tube radius

R_x: Bubble radius

R_{01}: inner tube radius

σ: Fluid surface tension

τ_{rz}: shear stress

τ_{wall}: shear stress at the wall

$\tau_{1/2}$: value of shear stress at which the apparent viscosity has dropped to half its zero shear value

τ^*: shear-stress level in relation to the transition between the Newtonian and shear-thinning regimes

T_b: temperature sensitivity coefficient

T_B: temperature of bulk fluid

T_i: initial temperature of polymer melt

T_p: temperature of polymer

T_s: temperature of steel/water interface

T_w: temperature of the wall

u_z: Fluid velocity

u^*: parabolic velocity profile

U_b: Bubble velocity

9 Appendix B: MATLAB code for PC Modeling


```
%*************************************************************************
%
%   PROJECT:          GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME:     Cross-Exp-ZSV_Model_PC
%   PURPOSE:          m vs Fo for PC
%   DATE:             10/25/2017
%
%*************************************************************************
clear variables
% close all
clc

%% Initial Conditions

% Number of nodes
nodes = 21;

% time measurements
t = linspace(1E-10,50,1000); %sec

% time increments
dt = 0.0001;

% Temperature of mold
Tb = (115.6-32)/1.8; %degC

% Temperature of melt
Ti = (623.6-32)/1.8; %degC

% Temperature of cooling fluid
Tc = 7; %degC

%Fractional coverage asymptotic value
m_n = 0.6;

%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m

% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
```

```
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s

% Step change in tube
dr2 = (R02 - R01) / (nodes - 1);

%% Physical Properties of the polymer

% Specific Heat
Cp_p = 2100; %J/(kg*K)

% Thermal Conductivity
k_p = 0.15; %W/(m*K)

% Density
den_p = 1017; %kg/m^3

% Thermal Diffusivity
a_p = k_p / (den_p * Cp_p); %m^2/s

% Temperature Sensitivity Coefficient
T_b = 1.1376E4; %K

% Zero shear viscosity
zero_visc = 1.0242*10^-4*exp(9.0679E3/(Ti+273.13));

% Frequency factor
B = zero_visc / exp(T_b / (25+273.13)); %kg/(m*s)

% Pressure gradient
delta_p = 2.42 * 10^6; %Pa

% n parameter
n = 1 - 0.03727;

% Step change in polymer
dr1 = R01 / (nodes - 1);

n_dr1 = 1.0 / (nodes - 1);

%% Physical Properties of the cooling fluid

% Specific Heat
Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)
```

```
% Viscosity
u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/(m*s)

% Density
den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

% Thermal Conductivity
k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

% Coefficient of Volume Expansion
b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

% Thermal Diffusivity
a_f = k_f / (den_f * Cp_f); %m^2/s

%% Heat Transfer outside the tube wall

% Initializing temperature profiles
Temp_p = zeros(nodes,2);
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Tc
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;

% Reynolds Number
Re = 1;

% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);
```

```
% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
        count = count + dt;

        % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
        i=nodes;
        j=1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+(dr2/2))↙
/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));

        % Temperature profile in Solid
        for i = nodes-1 : -1 : 2

            Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)*Temp_s↙
(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

        end

        % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
        i = 1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

        Temp_p(nodes,j+1) = Temp_s(i,j+1);

        % Temperature profile in Polymer
        for i = nodes-1 : -1 : 2

            Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p(i)↙
*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)*Temp_p↙
(i+1,j));

        end

        % Boundary Condition 5: @r=0, kdT/dr = 0
        i=1;
        Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));

        for i = 1 : nodes

            Temp_s(i,j) = Temp_s(i,j+1);
            Temp_p(i,j) = Temp_p(i,j+1);
```

```
        end

    end

    % Creating Temperature Profiles for Polymer and Solid
    for i = 1 : nodes

        Temp_fp(i,k) = Temp_p(i,j);
        Temp_fs(i,k) = Temp_s(i,j);

    end
end

%% Temperature Profile
figure
plot(Temp_fp(1:21,1),r_p)
hold on
plot(Temp_fs(1:21,1),r_s)
plot(Temp_fp(1:21,1),-r_p)
plot(Temp_fs(1:21,1),-r_s)
plot(Temp_fp(1:21,500),r_p)
plot(Temp_fs(1:21,500),r_s)
plot(Temp_fp(1:21,500),-r_p)
plot(Temp_fs(1:21,500),-r_s)
plot(Temp_fp(1:21,1000),r_p)
plot(Temp_fs(1:21,1000),r_s)
plot(Temp_fp(1:21,1000),-r_p)
plot(Temp_fs(1:21,1000),-r_s)

xlabel('Temperature (degC)')
ylabel('Radial Position (m)')
axis([0 350 -0.15 0.15])
% axis([0 350 -0.00635 0.00635])
% legend('t=0s','t=0s','t=25s','t=25s','t=50s','t=50s')
% legend boxon

%% Velocity Profile Calculation

%initialize velocity matrix
vel = zeros(nodes,length(t));

%velocity profile calculation
for k = 1 : length(t)

    %equation 4.33
    vel(nodes,k) = 0;

    for i = nodes-1 : -1 : 1

        f(i) = (n_dr1/2)*(delta_p/(2*L*B))^(1/n)*(((r_p(i+1)/exp(T_b/(Temp_fp(i+1,k)+273.13))))^↙
(1/n)+((r_p(i)/exp(T_b/(Temp_fp(i,k)+273.13))))^(1/n));
        vel(i,k) = vel(i+1,k) + f(i);

    end
```

```
end

%normalizing the velocity profile
n_vel = zeros(nodes,length(t));

for k = 1 : length(t)

    for i = 1 : nodes

        n_vel(i,k) = vel(i,k) / vel(1,k);

    end

end

%normalizing the radial position
n_radius = zeros(2*nodes,1);

for i = 1 : nodes
    n_radius(i) = r_p(i) / R01;
end

for i = 1 : nodes
    n_radius(i+nodes) = r_s(i) / R01;
end

%% plot velocity profile
figure
hold on
plot(n_vel(1:21,1),n_radius(1:21,1))
plot(n_vel(1:21,20),n_radius(1:21,1))
plot(n_vel(1:21,40),n_radius(1:21,1))
plot(n_vel(1:21,60),n_radius(1:21,1))
plot(n_vel(1:21,80),n_radius(1:21,1))
plot(n_vel(1:21,100),n_radius(1:21,1))
plot(n_vel(1:21,1),-n_radius(1:21,1))
plot(n_vel(1:21,20),-n_radius(1:21,1))
plot(n_vel(1:21,40),-n_radius(1:21,1))
plot(n_vel(1:21,60),-n_radius(1:21,1))
plot(n_vel(1:21,80),-n_radius(1:21,1))
plot(n_vel(1:21,100),-n_radius(1:21,1))
ylabel('Normalized Radius')
xlabel('Normalized Velocity')
legend('t = 1 s', 't = 20 s','t = 40 s','t = 60 s','t = 80 s','t = 100 s')
legend boxon
box on

%% Fractional Coverage Calculation

%initialize vectors
m = zeros(length(t),1);
Rx = zeros(length(t),1);

for k = 1 : length(t)

    %initializing summation variables
```

```
    sum = 0;

    for i = 1 : nodes-1

        %integrate through ru*
        sum = sum + (n_dr1 / 2) * (n_radius(i)*n_vel(i,k) + n_radius(i+1)*n_vel(i+1,k));

    end

    %integrate through ru*
    Rx(k) = R01 * sqrt(4 * sum);

    %Fractional coverage when integrating through ru*
    m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2-0.033;

end

%% Experimental Data

Fo_exp = [0.024058062   0.023212858 0.022752786 0.022245284 0.021805132 0.021334626 0.020703806
0.020302547 0.020025555 0.0188417   0.018269352 0.017706224 0.017191133 0.016183945 0.01573408
0.015466612 0.012500676 0.009558113 0.00661555  0.003672987 0.000730423 0   0.020624882
0.019661766 0.019155687 0.018606921 0.018137173 0.017640106 0.016980353 0.016564012 0.016277914
0.015378639 0.015065221 0.014483857 0.013914472 0.013395682 0.012386157 0.011937111 0.011670626
0.009928067 0.007668498 0.005408928 0.003149358 0.017492219 0.016570842 0.016083735 0.015553941
0.015099371 0.014617481 0.0139767   0.013571742 0.013293233 0.012416725 0.011543191 0.0104793
0.009491024 0.009051085 0.008789912 0.007908209 0.020522433 0.019524883 0.019010741 0.017989666
0.017496393 0.016845652 0.016436994 0.016156967 0.015280459 0.014976147 0.014413088 0.013863248
0.013363524 0.011964147 0.011709269 0.009561907 0.007387712 0.021607254 0.020811092 0.020380901
0.019908023 0.019498986 0.019062629 0.018478765 0.018107956 0.017852213 0.017044005 0.016760942
0.016234217 0.015716436 0.015243179 0.014318649 0.01390603  0.013660793 0.010888053 0.008366669
0.005845285 0.003323901 0.000802517 0.020658273 0.019731678 0.019280144 0.018809637 0.018419572
0.018017365 0.017497531 0.017176524 0.016958725 0.016287115 0.016057174 0.015635631 0.015228475
0.014861936 0.014159454 0.013851105 0.013669217 0.01130354  0.008685399 0.003449116 0.000830975
0.021628882 0.02085359  0.020432885 0.019969493 0.019568045 0.019139276 0.018564898 0.018199781
0.017947832 0.017151007 0.016351844 0.015840513 0.015372947 0.014459042 0.013808396 0.02165772
0.021021208 0.020655996 0.020243354 0.019879091 0.019484473 0.018948513 0.01860417  0.018365122
0.017602447 0.017333044 0.016828994 0.016330371 0.015872197 0.014971253 0.01456692  0.011614682
0.008973774 0.003691959 0.001051051 0   0.02060591  0.019497753 0.018948513 0.018370814
0.017888165 0.017387304 0.016735614 0.016330941 0.016055467 0.015201725 0.014908038 0.014367927
0.013844276 0.013371303 0.012059955 0.011822979 0.009816132 0.00765332  0.005490508 0.003327695
0.001164883];
m_exp = [0.697294689 0.724263224 0.723513367 0.735492284 0.723607207    0.724449991 0.727618456
0.717437048 0.726801285 0.725024622 0.720672799 0.728904796 0.720264006 0.722674305 0.718447447
0.718957268 0.70168097  0.66145734  0.648457964 0.647927231 0.624885193 0.596755298 0.670739472
0.703137029 0.691550379 0.698364593 0.701241289 0.702084104 0.702734839 0.695427813 0.700071162
0.69118994  0.697962535 0.696915367 0.695680165 0.697149418 0.685426476 0.697779619 0.691462878
0.652087133 0.645164384 0.62876281  0.619778545 0.65467158  0.665676326 0.665864314 0.663917007
0.665206569 0.66760097  0.664453562 0.659576701 0.653769429 0.648133579 0.655552792 0.651622842
0.648423281 0.648282304 0.633809094 0.628643675 0.696792714 0.704865477 0.707882141 0.708855373
0.705523156 0.699458032 0.701641672 0.704577476 0.699915757 0.702106235 0.702440258 0.68981505
0.70152703  0.676931134 0.637488469 0.623183955 0.613461083 0.68239709  0.69807312  0.707196133
0.706231596 0.707991626 0.70932566  0.708581543 0.703773498 0.708160712 0.690731214 0.703222289
0.703554796 0.702598996 0.702197434 0.692279374 0.697874714 0.697550677 0.660354829 0.656144062
0.640846623 0.634127931 0.633314738 0.670580778 0.701208446 0.702841462 0.701430498 0.707964274
0.704267557 0.704437699 0.697208985 0.700352376 0.689056354 0.698702575 0.701758078 0.698799327
```

```
0.699270899 0.686868841 0.695377846 0.692602561 0.628149629 0.622848174 0.61980335  0.616109316↙
0.672382415 0.699464312 0.701181268 0.701738315 0.703221545 0.705989659 0.701207358 0.700674887↙
0.703816372 0.698529973 0.706125555 0.704326992 0.708604945 0.682532306 0.673952105 0.70186614 ↙
0.704966546 0.715469093 0.709629477 0.711614645 0.705424738 0.704353297 0.710230389 0.708774738↙
0.703280722 0.711482142 0.712119017 0.714311394 0.710045123 0.705971812 0.703328536 0.699779951↙
0.707333813 0.662368545 0.641571248 0.637470269 0.682723977 0.704219021 0.699624337 0.701748647↙
0.703458037 0.70243286  0.704290352 0.697302691 0.698056443 0.696005632 0.694936722 0.698370737↙
0.698107561 0.69501785  0.682227046 0.681293654 0.641569264 0.613927755 0.610523635 0.610874927↙
0.60715379];


%% Fourier Number

Fo = a_p*t/R01^2;

%% Plot m vs Fo

figure
semilogx(Fo, m, Fo_exp, m_exp,'rs')
xlabel('Fo')
ylabel('m')
% title('Polycarbonate Model')
legend('Simulated Data','Experimental Data')
axis([1E-4 1E-1 0.55 0.95])
box on
legend boxoff
legend('Location','NorthWest')

%*************************************************************************
%
%                               CODE END
%
%*************************************************************************
```

10 Appendix C: MATLAB code for PC Simulation Comparison


```
%*************************************************************************
%
%   PROJECT:          GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME:     CE_ZSV_PC_Sim_vs_Exp.m
%   PURPOSE:          m vs Fo for PC
%   DATE:             10/26/2017
%
%*************************************************************************
clear variables
close all
clc

%% Load Excel Files

% delay time data references
% data_points = [1 23 44 9 31 52 11 34 54 21];
% Ti_dp = [346.35 348.25 351.15 347.65 349.05 351.75 346.82 348.645 350.942 347.65]; %degC
% gas pressure data references
data_points = [156 23 135 164 31 143 167 34 146 176 154];
Ti_dp = [348.27 347.25 347.54 349.21 349.05 348 348.721 348.645 347.362 348.85 348.83]; %degC
% shot size data references
% data_points = [120 23 99 128 31 107 130 34 110 119 ];
% Ti_dp = [347.55 348.25 348.25 348.15 349.05 348.65 348.69 348.645 347.64 347.45]; %degC
% piston speed data references
% data_points = [60 23 77 67 31 85 70 34 88 98];
% Ti_dp = [348.35 348.25 347.55 349.15 349.05 348.15 347.585 348.645 347.955 347.55]; %degC

for data_point_no = 1:length(data_points)
% find files
convertedrestime = xlsread('Fractional_Coverage_Comparison.xlsx','G2:G177');
convertedrowlength = xlsread('Fractional_Coverage_Comparison.xlsx','K2:K177');
factorr = xlsread('Fractional_Coverage_Comparison.xlsx','J2:J177');

dp = data_points(data_point_no);

converted_res_time=zeros(length(data_points),1);
converted_rowlength=zeros(length(data_points),1);
factor=zeros(length(data_points),1);

for i = 1:length(data_points)
    dp = data_points(i);
    converted_res_time(i) = convertedrestime(dp);
    converted_rowlength(i) = convertedrowlength(dp);
    factor(i) = factorr(dp);
end

%% Initial Conditions

% Number of nodes
nodes = 21;

% time measurements
%t = [5 10 20 30 40 50 60 90 120 180 240 480]; %sec
t = linspace(1E-10,50,1000); %sec

% time increments
```

```
dt = 0.0001;

% Temperature of mold
Tb = (115.6-32)/1.8; %degC

% Temperature of cooling fluid
Tc = 7; %degC

% Temperature of melt
Ti = Ti_dp(data_point_no);) %degC

%Fractional coverage asymptotic value
m_n = 0.6;

%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m

% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s

% Step change in tube
dr2 = (R02 - R01) / (nodes - 1);

%% Physical Properties of the polymer

% Specific Heat
Cp_p = 2100; %J/(kg*K)
```

```
% Thermal Conductivity
k_p = 0.15; %W/(m*K)

% Density
den_p = 1017; %kg/m^3

% Thermal Diffusivity
a_p = k_p / (den_p * Cp_p); %m^2/s

% Flow Activation Energy
delta_H_R = 1.1376E4; %K

% Zero shear viscosity
zero_visc = 1.0242*10^-4*exp(9.0679E3/(Ti+273.13));

% Frequency factor
A = zero_visc / exp(delta_H_R / (25+273.13)); %kg/(m*s)

% Pressure gradient
delta_p = 2.42 * 10^6; %Pa

% n parameter
n = 1 - 0.03727;

% Step change in polymer
dr1 = R01 / (nodes - 1);

n_dr1 = 1.0 / (nodes - 1);

%% Physical Properties of the cooling fluid

% Specific Heat
Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)

% Viscosity
u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/(m*s)

% Density
den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

% Thermal Conductivity
k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

% Coefficient of Volume Expansion
b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

% Thermal Diffusivity
a_f = k_f / (den_f * Cp_f); %m^2/s

%% Heat Transfer outside the tube wall

% Initializing temperature profiles
Temp_p = zeros(nodes,2);
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
```

```
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Ti
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;

% Reynolds Number
Re = 1;

% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);

% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
        count = count + dt;

        % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
        i=nodes;
        j=1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+(dr2/2))↙
/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));
```

```
        % Temperature profile in Solid
        for i = nodes-1 : -1 : 2

            Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)*Temp_s↙
(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

        end

        % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
        i = 1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

        Temp_p(nodes,j+1) = Temp_s(i,j+1);

        % Temperature profile in Polymer
        for i = nodes-1 : -1 : 2

            Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p(i)↙
*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)*Temp_p↙
(i+1,j));

        end

        % Boundary Condition 5: @r=0, kdT/dr = 0
        i=1;
        Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));

        for i = 1 : nodes

            Temp_s(i,j) = Temp_s(i,j+1);
            Temp_p(i,j) = Temp_p(i,j+1);

        end

    end

    % Creating Temperature Profiles for Polymer and Solid
    for i = 1 : nodes

        Temp_fp(i,k) = Temp_p(i,j);
        Temp_fs(i,k) = Temp_s(i,j);

    end
end

%% Velocity Profile Calculation

%initialize velocity matrix
vel = zeros(nodes,length(t));

%velocity profile calculation
for k = 1 : length(t)
```

```
    vel(nodes,k) = 0;

    for i = nodes-1 : -1 : 1

        f(i) = (n_dr1/2)*(delta_p/(2*L*A))^(1/n)*(((r_p(i+1)/exp(delta_H_R/(Temp_fp(i+1,k)+273.↙
13))))^(1/n)+((r_p(i)/exp(delta_H_R/(Temp_fp(i,k)+273.13))))^(1/n));
        vel(i,k) = vel(i+1,k) + f(i);

    end

end

%normalizing the velocity profile
n_vel = zeros(nodes,length(t));

for k = 1 : length(t)

    for i = 1 : nodes

        n_vel(i,k) = vel(i,k) / vel(1,k);

    end

end

%normalizing the radial position
n_radius = zeros(2*nodes,1);

for i = 1 : nodes
    n_radius(i) = r_p(i) / R01;
end

for i = 1 : nodes
    n_radius(i+nodes) = r_s(i) / R01;
end

% plot velocity profile
figure(1)
hold on
plot(n_vel(1:21,1),n_radius(1:21,1))
plot(n_vel(1:21,20),n_radius(1:21,1))
plot(n_vel(1:21,40),n_radius(1:21,1))
plot(n_vel(1:21,60),n_radius(1:21,1))
plot(n_vel(1:21,80),n_radius(1:21,1))
plot(n_vel(1:21,100),n_radius(1:21,1))
ylabel('Normalized Radius')
xlabel('Normalized Velocity')

%% Fractional Coverage Calculation

%initialize vectors
m = zeros(length(t),1);
Rx = zeros(length(t),1);

for k = 1 : length(t)
```

```
    %initializing summation variables
    sum = 0;

    for i = 1 : nodes-1

        %integrate through ru*
        sum = sum + (n_dr1 / 2) * (n_radius(i)*n_vel(i,k) + n_radius(i+1)*n_vel(i+1,k));

    end

    %integrate through ru*
    Rx(k) = R01 * sqrt(4 * sum);

    %Fractional coverage when integrating through ru*
    m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2-0.03;

end

%% Experimental Data

Fo_exp = [0.024058062   0.023212858 0.022752786 0.022245284 0.021805132 0.021334626 0.020703806
0.020302547 0.020025555 0.0188417   0.018269352 0.017706224 0.017191133 0.016183945 0.01573408
0.015466612 0.012500676 0.009558113 0.00661555  0.003672987 0.000730423 0   0.020624882
0.019661766 0.019155687 0.018606921 0.018137173 0.017640106 0.016980353 0.016564012 0.016277914
0.015378639 0.015065221 0.014483857 0.013914472 0.013395682 0.012386157 0.011937111 0.011670626
0.009928067 0.007668498 0.005408928 0.003149358 0.017492219 0.016570842 0.016083735 0.015553941
0.015099371 0.014617481 0.0139767   0.013571742 0.013293233 0.012416725 0.011543191 0.0104793
0.009491024 0.009051085 0.008789912 0.007908209 0.020522433 0.019524883 0.019010741 0.017989666
0.017496393 0.016845652 0.016436994 0.016156967 0.015280459 0.014976147 0.014413088 0.013863248
0.013363524 0.011964147 0.011709269 0.009561907 0.007387712 0.021607254 0.020811092 0.020380901
0.019908023 0.019498986 0.019062629 0.018478765 0.018107956 0.017852213 0.017044005 0.016760942
0.016234217 0.015716436 0.015243179 0.014318649 0.01390603  0.013660793 0.010888053 0.008366669
0.005845285 0.003323901 0.000802517 0.020658273 0.019731678 0.019280144 0.018809637 0.018419572
0.018017365 0.017497531 0.017176524 0.016958725 0.016287115 0.016057174 0.015635631 0.015228475
0.014861936 0.014159454 0.013851105 0.013669217 0.01130354  0.008685399 0.003449116 0.000830975
0.021628882 0.02085359  0.020432885 0.019969493 0.019568045 0.019139276 0.018564898 0.018199781
0.017947832 0.017151007 0.016351844 0.015840513 0.015372947 0.014459042 0.013808396 0.02165772
0.021021208 0.020655996 0.020243354 0.019879091 0.019484473 0.018948513 0.01860417  0.018365122
0.017602447 0.017333044 0.016828994 0.016330371 0.015872197 0.014971253 0.01456692  0.011614682
0.008973774 0.003691959 0.001051051 0   0.02060591  0.019497753 0.018948513 0.018370814
0.017888165 0.017387304 0.016735614 0.016330941 0.016055467 0.015201725 0.014908038 0.014367927
0.013844276 0.013371303 0.012059955 0.011822979 0.009816132 0.00765332  0.005490508 0.003327695
0.001164883];

m_exp = [0.697294689 0.724263224 0.723513367 0.735492284 0.723607207    0.724449991 0.727618456
0.717437048 0.726801285 0.725024622 0.720672799 0.728904796 0.720264006 0.722674305 0.718447447
0.718957268 0.70168097  0.66145734  0.648457964 0.647927231 0.624885193 0.596755298 0.670739472
0.703137029 0.691550379 0.698364593 0.701241289 0.702084104 0.702734839 0.695427813 0.700071162
0.69118994  0.697962535 0.696915367 0.695680165 0.697149418 0.685426476 0.697779619 0.691462878
0.652087133 0.645164384 0.62876281  0.619778545 0.65467158  0.665676326 0.665864314 0.663917007
0.665206569 0.66760097  0.664453562 0.659576701 0.653769429 0.648133579 0.655552792 0.651622842
0.648423281 0.648282304 0.633809094 0.628643675 0.696792714 0.704865477 0.707882141 0.708855373
0.705523156 0.699458032 0.701641672 0.704577476 0.699915757 0.702106235 0.702440258 0.68981505
0.70152703  0.676931134 0.637488469 0.623183955 0.613461083 0.68239709  0.69807312  0.707196133
0.706231596 0.707991626 0.70932566  0.708581543 0.703773498 0.708160712 0.690731214 0.703222289
```

```
0.703554796 0.702598996 0.702197434 0.692279374 0.697874714 0.697550677 0.660354829 0.656144062↙
0.640846623 0.634127931 0.633314738 0.670580778 0.701208446 0.702841462 0.701430498 0.707964274↙
0.704267557 0.704437699 0.697208985 0.700352376 0.689056354 0.698702575 0.701758078 0.698799327↙
0.699270899 0.686868841 0.695377846 0.692602561 0.628149629 0.622848174 0.61980335  0.616109316↙
0.672382415 0.699464312 0.701181268 0.701738315 0.703221545 0.705989659 0.701207358 0.700674887↙
0.703816372 0.698529973 0.706125555 0.704326992 0.708604945 0.682532306 0.673952105 0.70186614↙
0.704966546 0.715469093 0.709629477 0.711614645 0.705424738 0.704353297 0.710230389 0.708774738↙
0.703280722 0.711482142 0.712119017 0.714311394 0.710045123 0.705971812 0.703328536 0.699779951↙
0.707333813 0.662368545 0.641571248 0.637470269 0.682723977 0.704219021 0.699624337 0.701748647↙
0.703458037 0.70243286  0.704290352 0.697302691 0.698056443 0.696005632 0.694936722 0.698370737↙
0.698107561 0.69501785  0.682227046 0.681293654 0.641569264 0.613927755 0.610523635 0.610874927↙
0.60715379];


%% Fourier Number

Fo = a_p*t/R01^2;

%% Plot m vs Fo
figure (2)
semilogx(Fo, m, Fo_exp, m_exp,'rs')
xlabel('Fo')
ylabel('m')
axis([1E-5 1E-1 0.55 0.95])
title('Polycarbonate Model')
legend('Simulated Data','Experimental Data')

%% Determination of shear_rate

% turn converted_res_time and converted_rowlength into integers
for i = 1:length(converted_rowlength)
   converted_rowlength(i) = round(converted_rowlength(i));
end
for i = 1:length(converted_res_time)
   converted_res_time(i) = round(converted_res_time(i));
end

% calculate n_vel_sim
n_vel_sim = zeros(length(converted_rowlength),length(converted_res_time));
for i = 1:length(converted_rowlength)
    if converted_rowlength(i) < 101
        if converted_res_time(i) <101
            n_vel_sim(i,i) = n_vel(converted_rowlength(i),converted_res_time(i));
        else
            n_vel_sim(i,i) = 0;
        end
    else
        n_vel_sim(i,k) = 0;
    end
end

% non_normalize n_vel_sim
vel_sim = zeros(length(converted_rowlength),1);
for i = 1:length(converted_rowlength)
   vel_sim(i,1) = n_vel_sim(i,i)*factor(i);
end
```

```
% determine m_sim
m_sim = zeros(length(converted_res_time),1);
for i = 1:length(converted_res_time)
    if converted_res_time(i) < 1001
   m_sim(i) = m(converted_res_time(i));
    else
        m_sim(i) = 0;
    end
end

% calculate shear rate (gamma_dot)
vel_diff = zeros(length(vel_sim),1);
t_diff = zeros(length(vel_sim),1);
n_gamma_dot = zeros(length(vel_sim),1);
gamma_dot = zeros(length(vel_sim),1);
for i =1:length(vel_sim)
    if converted_res_time(i) < 1001
        vel_diff(i) = n_vel(converted_rowlength(i)-1,converted_res_time(i))-n_vel
(converted_rowlength(i),converted_res_time(i));
        t_diff(i) = 0.05;
        n_gamma_dot(i) = vel_diff(i)/t_diff(i);
        gamma_dot(i) = n_gamma_dot(i)*factor(i);
    else
        gamma_dot(i)=0;
    end
end

%% Plot Viscosity vs shear rate with the newly acquired shear rate values
% Method 1: Cross-exponential Model
lenn = 1E2;
Ti = Ti+273.15; %K
m_power = A.*exp(delta_H_R./Ti); %Pa-s
shear_rate = logspace(-3, 10, lenn);
n_p = 0.03727;
B = 1.0242E-4;
Tb = 9.0679E3;
no = B.*exp(Tb./Ti);
tau_star = 9.4221E5;
for k = 1:length(shear_rate)
    for i = 1:length(Ti)
        viscosity(i,k) = no(i)/(1+(no(i)*shear_rate(k)/tau_star)^(1-n_p));
        visc_dp(data_point_no,k) = viscosity(i,k);
    end
end
figure (3)
loglog(shear_rate, viscosity(1,1:lenn))
hold on
% loglog(shear_rate, viscosity(2,1:lenn))
% loglog(shear_rate, viscosity(3,1:lenn))
% loglog(shear_rate, viscosity(4,1:lenn))
% loglog(shear_rate, viscosity(5,1:lenn))
% loglog(shear_rate, viscosity(6,1:lenn))
% loglog(shear_rate, viscosity(7,1:lenn))
% loglog(shear_rate, viscosity(8,1:lenn))
% loglog(shear_rate, viscosity(9,1:lenn))
```

```
% loglog(shear_rate, viscosity(10,1:lenn))

% plot data points
n_p = 0.03727;
B = 1.0242E-4;
Tb = 9.0679E3;
no_exp = B*exp(Tb/Ti);
tau_star = 9.4221E5;
for i = 1:length(gamma_dot)
    visc_exp(i) = no_exp/(1+(no_exp*gamma_dot(i)/tau_star)^(1-n_p));
end
visc_exp_dp(data_point_no) = visc_exp(1);
loglog(gamma_dot(1), visc_exp(1), 'rs')
leg_ent = ['ABCDEFGHIJKLMN'];
legend(leg_ent(data_point_no))
% loglog(gamma_dot(2), visc_exp(2), 'bo')
% loglog(gamma_dot(3), visc_exp(3), 'gs')
% legend('high delay time', 'base delay time', 'low delay time')
xlabel('\gamma_{dot} [1/s]')
ylabel('\eta [Pa-s]')
title('CE Model: PC')
hold off

% CONCLUSION: All shear-rates tested are before elbow region.  Therefore,
% assumption that fluid behavior is Newtonian is valid for all points

%% Plot m_sim vs m_exp
figure (4)
m_exp = [m_exp(data_points(data_point_no))];
m_exp_dp(data_point_no) = m_exp;
m_sim_dp(data_point_no) = m_sim(data_point_no);
plot(m_sim(1), m_exp(1),'rs')
xlabel('m_{sim}')
ylabel('m_{exp}')
leg_ent = ['ABCDEFGHIJKLMN'];
legend(leg_ent(data_point_no))
title('PC- Rowlength: 90mm')
axis([0.5 1 0.5 1])
hold on
line = linspace(0.5,1,100);
plot(line,line)

end

%% PLOT RESULTS
figure(100)
loglog(shear_rate, visc_dp(1,1:lenn))
marker_point = ['s','>','x','s','>','x','s','>','x','s','>','x'];
 hold on
 for i = 2:length(visc_exp_dp)
 loglog(shear_rate, visc_dp(i,1:lenn))
 end
 for i = 1:length(visc_exp_dp)
 loglog(gamma_dot(i), visc_exp_dp(i), marker_point(i))
 end
  title('Varying Pre-charge Gas Pressure Setting')
```

```
legend('High 90mm','Base 90mm','Low 90mm','High 285mm','Base 285mm','Low 285mm','High↙
343mm','Base 343mm','Low 343mm','High 505mm','Low 505mm','High 90mm','Base 90mm','Low 90mm','High↙
285mm','Base 285mm','Low 285mm','High 343mm','Base 343mm','Low 343mm','High 505mm','Low 505mm')
legend boxoff
box on
legend('Location','SouthWest')
set(legend,'FontSize',7)
xlabel('\gamma_{dot} [1/s]')
ylabel('\eta [Pa-s]')
 hold off

 figure(200)
 hold on
 for i = 1:length(m_sim_dp)
     plot(m_sim_dp(i), m_exp_dp(i),marker_point(i))
 end
 line = linspace(0.5,1,100);
plot(line,line)
 xlabel('m_{sim}')
ylabel('m_{exp}')
legend('High 90mm','Base 90mm','Low 90mm','High 285mm','Base 285mm','Low 285mm','High↙
343mm','Base 343mm','Low 343mm','High 505mm','Low 505mm','Perfect Simulation');
title('Varying Pre-charge Gas Pressure Setting')
axis([0.5 1 0.5 1])
box on
legend boxoff
legend('Location','NorthWest')
%*************************************************************************
%
%                                CODE END
%
%*************************************************************************
```

11 Appendix D: MATLAB code for PS: Cross-Exponential Model


```
%*************************************************************************
%
%   PROJECT:         GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME: Kaminski_Implementation_of_Cross-exponential_Model
%   PURPOSE:         m vs Fo for PS
%   DATE:            10/27/2017
%   DESCRIPTION: This code implements the Cross-exponential model and
%   solves the differential equation obtained for the velocity profile with
%   the use of the shooting method.
%
%*************************************************************************
clear all
clc
%% Initial Conditions

% Number of nodes
nodes = 157; %this number is changed from 21 to 157 because the Mathematica solution gives 157↙
data points

% time measurements
t = linspace(1E-10,50,1000); %sec

% time increments
dt = 0.0001;

% Temperature of mold
Tb = (180-32)/1.8; %degC

% Temperature of melt
Ti = (485.6-32)/1.8; %degC

    % Temperature of cooling water
    Tc = 25; %degC

%Fractional coverage asymptotic value
n_p = 0.2968; %n parameter for Cross model
if n_p >= 0.3
    m_n = 0.1516*log(n_p)+0.7259; %formulas obtained from Poslinski & Coyle
else
    m_n = -0.1428*n_p^2+0.2626*n_p+0.4782;
end

%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m
```

```
% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s

% Step change in tube
dr2 = (R02 - R01) / (nodes - 1);

%% Physical Properties of the polymer

% Specific Heat
Cp_p = 2100; %J/(kg*K)

% Thermal Conductivity
k_p = 0.15; %W/(m*K)

% Density
den_p = 940; %kg/m^3

% Thermal Diffusivity
a_p = k_p / (den_p * Cp_p); %m^2/s

% Flow Activation Energy
delta_H_R = 9.0679E3; %K

% Zero shear viscosity
zero_visc = 1.6783*10^-6*exp(1.1376*10^4/(Ti+273.13));

% Frequency factor
A = zero_visc / exp(delta_H_R / (25+273.13)); %kg/(m*s)

% Pressure gradient
delta_p = 2.42 * 10^6; %Pa

% Step change in polymer
dr1 = R01 / (nodes - 1);

n_dr1 = 1.0 / (nodes - 1);

%% Physical Properties of the cooling fluid
```

```
% Specific Heat
Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)

% Viscosity
u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/(m*s)

% Density
den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

% Thermal Conductivity
k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

% Coefficient of Volume Expansion
b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

% Thermal Diffusivity
a_f = k_f / (den_f * Cp_f); %m^2/s

%% Heat Transfer outside the tube wall

% Initializing temperature profiles
Temp_p = zeros(nodes,2);
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Tc
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;

% Reynolds Number
Re = 1;
```

```
% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);

% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
        count = count + dt;

        % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
        i=nodes;
        j=1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+(dr2/2))↙
/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));

        % Temperature profile in Solid
        for i = nodes-1 : -1 : 2

            Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)*Temp_s↙
(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

        end

        % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
        i = 1;
        Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

        Temp_p(nodes,j+1) = Temp_s(i,j+1);

        % Temperature profile in Polymer
        for i = nodes-1 : -1 : 2

            Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p(i)↙
*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)*Temp_p↙
(i+1,j));

        end

        % Boundary Condition 5: @r=0, kdT/dr = 0
        i=1;
        Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));
```

```
        for i = 1 : nodes

            Temp_s(i,j) = Temp_s(i,j+1);
            Temp_p(i,j) = Temp_p(i,j+1);

        end

    end

    % Creating Temperature Profiles for Polymer and Solid
    for i = 1 : nodes

        Temp_fp(i,k) = Temp_p(i,j);
        Temp_fs(i,k) = Temp_s(i,j);

    end
end

%% Temperature Profile
figure
ai = Temp_fp(1:nodes,1);
bi = Temp_fp(1:nodes,20);
ci = Temp_fp(1:nodes,40);
di = Temp_fp(1:nodes,60);
ei = Temp_fp(1:nodes,80);
fi = Temp_fp(1:nodes,100);
plot(ai,r_p','b',bi,r_p', 'r', ci,r_p','g',di,r_p','c',ei,r_p','m',fi,r_p','y', ai(end:-1:1),↙
r_p'-0.00635, 'b', bi(end:-1:1),r_p'-0.00635,'r', ci(end:-1:1),r_p'-0.00635,'g', di(end:-1:1),↙
r_p'-0.00635,'c', ei(end:-1:1),r_p'-0.00635,'m', fi(end:-1:1),r_p'-0.00635,'y')
axis([50 275 -0.00635 0.00635 ])
xlabel('Temperature (^{o}C)')
ylabel('Radial Position (m)')
legend('t = 1 sec','t = 10 sec' ,'t = 20 sec','t = 30 sec','t = 40 sec','t = 50 sec')

%% Viscosity Model Parameters
n = 0.2968;
tao_star = 1.738E4; %Pa
B = 1.6783E-6; %Pa-s
Tb = 1.1376E4 - 273.15; %degC

%initialize eta_o
eta_o = zeros(nodes,length(t));
%calculate eta_o
for k = 1 : length(t)
    for i = 1 : nodes
        eta_o(i,k) = B.*exp(Tb./Temp_fp(i,k));
    end
end

%% Velocity Profile Calculation

% Loading data from Mathematica solution:
% Note: the Mathematica solution solves for the DIMENSIONLESS velocity

% define search directory
search_directory = 'Mathematica_solutions';
```

```
% find files in directory
files = Find_Files(search_directory);
files = Extract_Files('CE_Model_PS_data.txt', files);

filename = strcat(search_directory, filesep, files);
filename = char(filename);
fid = fopen(filename);
filedata = textscan(fid, '%f%f', 'Delimiter', ' ', 'Headerlines', 0);
fclose(fid);

dim_r_p = cell2mat(filedata(:,1)); %normalized r_p
dim_vel = cell2mat(filedata(:,2)); %normalized velocity

%plot dimensionless velocity profile
figure
plot(dim_vel, dim_r_p*0.00635);
hold on
plot(dim_vel, -dim_r_p*0.00635)
ylabel('Radius')
xlabel('Dimensionless Velocity')
title('Dimensionless Velocity Profile for CE Model for PS')
axis([0 1 -0.00635 0.00635])

% solving for normalized velocity... eventually
% but first, non-normalized velocity,
% initialize velocity matrix
vel = zeros(nodes,length(t));
% velocity profile calculation
for k = 1 : length(t)
    for i = nodes-1 : -1 : 1

        vel(i,k) = dim_vel(i)*delta_p*R01^2/(L*eta_o(i,k));

    end

end

% Plot Velocity Profile
figure
hold on
plot(vel(1:157,1),dim_r_p)
plot(vel(1:157,10),dim_r_p)
plot(vel(1:157,20),dim_r_p)
plot(vel(1:157,30),dim_r_p)
plot(vel(1:157,40),dim_r_p)
plot(vel(1:157,50),dim_r_p)
plot(vel(1:157,60),dim_r_p)
plot(vel(1:157,70),dim_r_p)
plot(vel(1:157,80),dim_r_p)
plot(vel(1:157,90),dim_r_p)
plot(vel(1:157,100),dim_r_p)
ylabel(' Normalized Radius')
xlabel('Velocity')
title('Velocity Profile for CE Model for PS')
```

```
% Now that we can calculate vel_max we can find normaliized velocity!
%normalizing the velocity profile
n_vel = zeros(nodes,length(t));

for k = 1 : length(t)

    for i = 1 : nodes

        n_vel(i,k) = vel(i,k) / vel(1,k);

    end

end

%% Plot Normalized Velocity Profile
figure
hold on
plot(n_vel(1:157,1),dim_r_p)
%plot(n_vel(1:157,10),dim_r_p)
plot(n_vel(1:157,20),dim_r_p)
%plot(n_vel(1:157,30),dim_r_p)
plot(n_vel(1:157,40),dim_r_p)
%plot(n_vel(1:157,50),dim_r_p)
plot(n_vel(1:157,60),dim_r_p)
%plot(n_vel(1:157,70),dim_r_p)
plot(n_vel(1:157,80),dim_r_p)
%plot(n_vel(1:157,90),dim_r_p)
plot(n_vel(1:157,100),dim_r_p)
plot(n_vel(1:157,1),-dim_r_p)
%plot(n_vel(1:157,10),dim_r_p)
plot(n_vel(1:157,20),-dim_r_p)
%plot(n_vel(1:157,30),dim_r_p)
plot(n_vel(1:157,40),-dim_r_p)
%plot(n_vel(1:157,50),dim_r_p)
plot(n_vel(1:157,60),-dim_r_p)
%plot(n_vel(1:157,70),dim_r_p)
plot(n_vel(1:157,80),-dim_r_p)
%plot(n_vel(1:157,90),dim_r_p)
plot(n_vel(1:157,100),-dim_r_p)
ylabel('Normalized Radius')
xlabel('Normalized Velocity')
title('Normalized Velocity Profile for CE Model for PS')
box on
legend('t=1 sec','t=20 sec','t=40 sec','t=60 sec','t=80 sec','t=100 sec')

%% Fractional Coverage Calculation

%initialize vectors
m = zeros(length(t),1);
Rx = zeros(length(t),1);
eta_avg = zeros(length(t),1);

% Rx calculation
for k = 1 : length(t)

    Rx(k) = (eta_o(1,k)*R01^4/eta_o(100,k))^(1/4);
```

```
    %Fractional coverage when integrating through ru*
    m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2;

end

%% Experimental Data

Fo_exp = [0.009290161   0.009361771 0.009319921 0.00923157  0.00912741  0.00878424  0.008636369
0.008528489 0.007097774 0.006588133 0.005670779 0.00480439  0.004039928 0   0   0   0.010646289
0.01123963  0.011360531 0.011388431 0.011343791 0.01123963  0.01102573  0.01085275  0.010718829
0.010227788 0.010034348 0.009548658 0.009240126 0.008843946 0.005331866 0.001957565 0.001186136
0.000718709 0.000435483 0.009763718 0.010065503 0.010091078 0.010046903 0.009961343 0.009831143
0.009605617 0.009436822 0.009310342 0.00869282  0.008356928 0.008006014 0.007669818 0.001254455
0.000748789 0.000446954 0.000266789 0.013721433 0.014074369 0.014123194 0.014102269 0.014035309
0.013923709 0.013566588 0.013449408 0.013030907 0.012869087 0.012549687 0.01221483  0.011892585
0.000900184 0.000436854 0.000212003 0.000102884 1.90411E-05 0.007187798 0.007228718 0.007172918
0.007070618 0.006955298 0.006810218 0.006430777 0.006317317 0.005789076 0.00551253  0.005229214
0.004961374 0.000574741 0   0   0.009706057 0.010012958 0.010078058 0.010096658 0.010078058
0.010029698 0.009843698 0.009778598 0.009538657 0.0092543   0.00905196  0.008857896 0.001638369
0.000986789 0.000594343 0.000357973 0.010386076 0.01071855  0.010758076 0.01072785  0.01065345
0.01053441  0.010322835 0.010162409 0.009446308 0.009120212 0.008779032 0.008451206 0.000800971
0.00035954  0.00016139  7.24447E-05 0.010082522 0.010394538 0.010424763 0.010385238 0.010303398
0.010176917 0.009790037 0.009665416 0.009055335 0.008722543 0.008375039 0.008041633 0.002343016 0
0.009265981 0.009445006 0.009414781 0.009314806 0.009184605 0.009009765 0.008526164 0.008377364
0.007670562 0.007294246 0.006905696 0.00653596  0.000510563 0.000135167 6.95478E-05 0.009419617
0.009409387 0.009330337 0.009204786 0.009070866 0.008907186 0.008490545 0.008367785 0.007802344
0.007509058 0.007209932 0.006928142 0.001532225 0.000979925 0.000626705 0.000400805 0.000256333
0.009973712 0.010556823 0.010673073 0.010696323 0.010647963 0.010540083 0.010321533 0.010145762
0.010009982 0.009513361 0.009318061 0.008928142 0.008517329 0.008118358 0.000379347 0.000187439
9.26151E-05 1.78759E-05];
m_exp = [0.664455133    0.660600195 0.659903584 0.653815611 0.643863879 0.643420578 0.6420786
0.641032095 0.63    0.6320329   0.630065371 0.6241699   0.61863233  0.527927299 0.511768978
0.498364331 0.689057396 0.692621509 0.692304866 0.689900398 0.68720811  0.68546886  0.685555
0.684666754 0.687869965 0.680615113 0.688518593 0.669690715 0.667162285 0.66521013  0.643249142
0.562980759 0.533264801 0.512741854 0.505587609 0.672333745 0.680039601 0.667376696 0.668113545
0.669887703 0.665519991 0.66042195  0.663392543 0.65819495  0.651606294 0.642900574 0.641887469
0.635561537 0.536369509 0.520796145 0.51029797  0.492728249 0.71303761  0.6994871   0.701769188
0.69999603  0.701257595 0.698408593 0.699423999 0.698485672 0.699962583 0.699467845 0.686372017
0.680490058 0.670616312 0.54996579  0.516916765 0.498031168 0.493316219 0.486415845 0.646852772
0.641527395 0.6402977   0.642550577 0.641688346 0.640829917 0.634794885 0.629447227 0.622278177
0.619749266 0.613333547 0.602327828 0.541584269 0.527157593 0.518919719 0.675408624 0.668449729
0.665198697 0.666624644 0.661579087 0.658701847 0.659627084 0.658785575 0.65831585  0.648579795
0.642704076 0.639065328 0.568488481 0.537996416 0.522383791 0.512449143 0.698873255 0.692935765
0.690407814 0.691673269 0.688984876 0.692774128 0.694173317 0.688442735 0.678472635 0.669053441
0.661023285 0.658290052 0.586277093 0.526014283 0.515202825 0.496022157 0.679814469 0.678385666
0.678443498 0.675245682 0.675144517 0.678017961 0.670001147 0.671366624 0.669771592 0.659192099
0.656571262 0.650316108 0.593335238 0.547726996 0.673297245 0.672809622 0.669223792 0.668933217
0.667955017 0.66403757  0.662242917 0.659374749 0.657487586 0.654273822 0.636189927 0.625708532
0.500466487 0.473828363 0.457995038 0.693084074 0.690318652 0.687879622 0.68319826  0.684136001
0.679103836 0.679561568 0.67289133  0.6701  0.66521 0.65222 0.6294  0.575210798 0.539712115
0.529919918 0.527618377 0.515450591 0.686706115 0.679323206 0.684293066 0.689236851 0.684060921
0.686483024 0.681287794 0.677176137 0.672125212 0.668292035 0.667041945 0.665278969 0.661918419
0.65875516  0.562243403 0.544177804 0.532056223 0.511946383];

%% Fourier Number
```

```
Fo = a_p*t/R01^2;

%% Plot Fo vs m
figure
semilogx(Fo, m, Fo_exp, m_exp,'rs')
xlabel('Fo')
ylabel('m')
title('Polystyrene Model')
axis([1E-5 1E-1 0.45 0.9])
legend('Simulated Data','Experimental Data')
%*************************************************************************
%
%                                CODE END
%
%*************************************************************************
```

12 Appendix E: MATLAB code for PS: Ellis Model


```
%************************************************************************
%
%   PROJECT:          GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME:     Ellis_Model
%   PURPOSE:          m vs Fo for PS
%   DATE:             10/27/2017
%
%************************************************************************
clear variables
close all
clc
%% Import Mathematica Solution
% Loading data from Mathematica solution:
% Note: the Mathematica solution solves for the DIMENSIONLESS velocity

% define search directory
search_directory = 'Mathematica_solutions';

% find files in directory
filess = Find_Files(search_directory);
for indexx=4:length(filess)
    files = Extract_Files(filess(indexx), filess);

    filename = strcat(search_directory, filesep, files);
    filename = char(filename);
    fid = fopen(filename);
    filedata = textscan(fid, '%f%f', 'Delimiter', ' ', 'Headerlines', 0);
    fclose(fid);
    %     factor = 140;
    if indexx == 4
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 7
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 8
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 9
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 5
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 6
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 10
```

```
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 11
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 12
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx ==13
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx ==14
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    end
    factor = round(0.9*length(dim_vel));
    dim_vel_plot(1:length(dim_vel),indexx) = dim_vel(1:length(dim_vel),1);
    dim_r_p_plot(1:length(dim_r_p),indexx) = dim_r_p(1:length(dim_r_p),1);
    %% Initial Conditions

    % Number of nodes
    nodes = length(dim_vel); %this number varies based on the Mathematica solution

    % time measurements
    t = linspace(1E-10,50,1000); %sec

    % time increments
    dt = 0.0001;

    % Temperature of mold
    Tb = (180-32)/1.8; %degC

    % Temperature of melt
    Ti = (623.6-32)/1.8; %degC

    % Temperature of cooling water
    Tc = 25; %degC

    %Fractional coverage asymptotic value
    n_p = 0.2968;
    if n_p >= 0.3
        m_n = 0.1516*log(n_p)+0.7259;
```

```
else
    m_n = -0.1428*n_p^2+0.2626*n_p+0.4782;
end

%% Ellis model parameters

% a: (measure of degree of shear thinning behavior)
a = 3.26;

% Tao_half: (value of shear stress at which the apparent
% viscosity has dropped to half its zero shear value)
tao_half = 16618.6;

% Zero Shear Viscosity
% function of temperature
% defined later

%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m

% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s

% Step change in tube
dr2 = (R02 - R01) / (nodes - 1);

%% Physical Properties of the polymer
```

```
    % Specific Heat
    Cp_p = 2100; %J/(kg*K)

    % Thermal Conductivity
    k_p = 0.15; %W/(m*K)

    % Density
    den_p = 940; %kg/m^3

    % Thermal Diffusivity
    a_p = k_p / (den_p * Cp_p); %m^2/s

    % Flow Activation Energy
    %delta_H_R = 8575; %K
    delta_H_R = 9.0679E3; %K

    % Zero shear viscosity
    zero_visc = 1*10^16*exp(-0.056*(Ti+273.13));

    % Frequency factor
    A = zero_visc / exp(delta_H_R / (25+273.13)); %kg/(m*s)

    % Pressure gradient
    delta_p = 2.42 * 10^6; %Pa

    % Step change in polymer
    dr1 = R01 / (nodes - 1);

    n_dr1 = 1.0 / (nodes - 1);

    %% Physical Properties of the cooling fluid

    % Specific Heat
    Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)

    % Viscosity
    u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/↙
(m*s)

    % Density
    den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

    % Thermal Conductivity
    k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

    % Coefficient of Volume Expansion
    b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

    % Thermal Diffusivity
    a_f = k_f / (den_f * Cp_f); %m^2/s

    %% Heat Transfer outside the tube wall

    % Initializing temperature profiles
    Temp_p = zeros(nodes,2);
```

```
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Tc
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;


% Reynolds Number
Re = 1;

% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);

% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
        count = count + dt;

        % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
        i=nodes;
        j=1;
```

```
            Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+↙
(dr2/2))/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));

            % Temperature profile in Solid
            for i = nodes-1 : -1 : 2

                Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)↙
*Temp_s(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

            end

            % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
            i = 1;
            Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

            Temp_p(nodes,j+1) = Temp_s(i,j+1);

            % Temperature profile in Polymer
            for i = nodes-1 : -1 : 2

                Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p↙
(i)*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)↙
*Temp_p(i+1,j));

            end

            % Boundary Condition 5: @r=0, kdT/dr = 0
            i=1;
            Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));

            for i = 1 : nodes

                Temp_s(i,j) = Temp_s(i,j+1);
                Temp_p(i,j) = Temp_p(i,j+1);

            end

        end

        % Creating Temperature Profiles for Polymer and Solid
        for i = 1 : nodes

            Temp_fp(i,k) = Temp_p(i,j);
            Temp_fs(i,k) = Temp_s(i,j);

        end
    end

    %% Temperature Profile Calculation
    % figure
    % plot(r_p,Temp_fp(1:nodes,4))
    % hold on
```

```
% plot(r_p,Temp_fp(1:nodes,5))
% plot(r_p,Temp_fp(1:nodes,7))

%% Velocity Profile Calculation

%initialize eta_o
eta_o = zeros(nodes,length(t));
%calculate eta_o
for k = 1 : length(t)
    for i = 1 : nodes
        eta_o(i,k) = 1*10^16*exp(-0.056*(Temp_fp(i,k)+273.13));
    end
end

% THE INPUT OF THE MATHEMATHICA SOLUTION IS IMPORTANT FOR THE FOLLOWING
% PORTION OF CODE

%plot dimensionless velocity profile
figure
plot(dim_vel, dim_r_p*0.00635);
hold on
plot(dim_vel, -dim_r_p*0.00635)
ylabel('Radius')
xlabel('Dimensionless Velocity')
title('Dimensionless Velocity Profile for Ellis Model for PS')
axis([0 1 -0.00635 0.00635])

% solving for normalized velocity... eventually
% but first, non-normalized velocity,
% initialize velocity matrix
vel = zeros(nodes,length(t));
% velocity profile calculation
for k = 1 : length(t)
    for i = nodes-1 : -1 : 1

        vel(i,k) = dim_vel(i)*delta_p*R01^2/(L*eta_o(i,k));

    end

end

% Plot Velocity Profile
figure
hold on
plot(vel(1:nodes,1),dim_r_p)
plot(vel(1:nodes,10),dim_r_p)
plot(vel(1:nodes,20),dim_r_p)
plot(vel(1:nodes,30),dim_r_p)
plot(vel(1:nodes,40),dim_r_p)
plot(vel(1:nodes,50),dim_r_p)
plot(vel(1:nodes,60),dim_r_p)
plot(vel(1:nodes,70),dim_r_p)
plot(vel(1:nodes,80),dim_r_p)
plot(vel(1:nodes,90),dim_r_p)
plot(vel(1:nodes,100),dim_r_p)
plot(vel(1:nodes,1),-dim_r_p)
```

```
plot(vel(1:nodes,10),-dim_r_p)
plot(vel(1:nodes,20),-dim_r_p)
plot(vel(1:nodes,30),-dim_r_p)
plot(vel(1:nodes,40),-dim_r_p)
plot(vel(1:nodes,50),-dim_r_p)
plot(vel(1:nodes,60),-dim_r_p)
plot(vel(1:nodes,70),-dim_r_p)
plot(vel(1:nodes,80),-dim_r_p)
plot(vel(1:nodes,90),-dim_r_p)
plot(vel(1:nodes,100),-dim_r_p)
ylabel(' Normalized Radius')
xlabel('Velocity')
title('Velocity Profile for CE Model for PS')

% Now that we can calculate vel_max we can find normaliized velocity!
% normalizing the velocity profile
n_vel = zeros(nodes,length(t));

for k = 1 : length(t)

    for i = 1 : nodes

        n_vel(i,k) = vel(i,k) / vel(1,k);

    end

end

% Plot Normalized Velocity Profile
figure
hold on
plot(n_vel(1:nodes,10),dim_r_p)
%plot(n_vel(1:157,10),dim_r_p)
plot(n_vel(1:nodes,200),dim_r_p)
%plot(n_vel(1:157,30),dim_r_p)
plot(n_vel(1:nodes,400),dim_r_p)
%plot(n_vel(1:157,50),dim_r_p)
plot(n_vel(1:nodes,600),dim_r_p)
%plot(n_vel(1:157,70),dim_r_p)
plot(n_vel(1:nodes,800),dim_r_p)
%plot(n_vel(1:157,90),dim_r_p)
plot(n_vel(1:nodes,1000),dim_r_p)
plot(n_vel(1:nodes,10),-dim_r_p)
%plot(n_vel(1:157,10),-dim_r_p)
plot(n_vel(1:nodes,200),-dim_r_p)
%plot(n_vel(1:157,30),-dim_r_p)
plot(n_vel(1:nodes,400),-dim_r_p)
%plot(n_vel(1:157,50),-dim_r_p)
plot(n_vel(1:nodes,600),-dim_r_p)
%plot(n_vel(1:157,70),-dim_r_p)
plot(n_vel(1:nodes,800),-dim_r_p)
%plot(n_vel(1:157,90),-dim_r_p)
plot(n_vel(1:nodes,100),-dim_r_p)
ylabel('Normalized Radius')
xlabel('Normalized Velocity')
title('Normalized Velocity Profile for Ellis Model for PS')
```

```
    box on
    legend('t=1 sec','t=20 sec','t=40 sec','t=60 sec','t=80 sec','t=100 sec')

end
%% Plot All Dimensionless Velocity Profiles
figure
plot(dim_vel_plot(1:173,4),dim_r_p_plot(1:173,4))
hold on
plot(dim_vel_plot(1:173,4),-dim_r_p_plot(1:173,4))
% title('\tau_{wall}=1.47E6 Pa')
ylabel('dimensionless radius')
xlabel('dimensionless velocity')
for i = 5:length(filess)
    plot(dim_vel_plot(1:173,i),dim_r_p_plot(1:173,i))
end
for i = 5:length(filess)
    plot(dim_vel_plot(1:173,i),-dim_r_p_plot(1:173,i))
end
title('All Dimensionless Velocity Profiles')

%% Fractional Coverage Calculation

    %initialize vectors
    m = zeros(length(t),1);
    Rx = zeros(length(t),1);

    %define shear stress average and wall
    tau_avg = linspace(1.47E6,3.42,length(t));
    tau_wall = tau_avg(1); %isothermal

    % Rx calculation
    for k = 1 : length(t)

        Rx(k) = R01*sqrt(tau_avg(k)*eta_o(1,k)/(tau_wall*eta_o(105,k)));

        %Fractional coverage when integrating through ru*
        m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2;

    end


    %% Experimental Data

    Fo_exp = [0.009290161   0.009361771 0.009319921 0.00923157  0.00912741  0.00878424↙
0.008636369 0.008528489 0.007097774 0.006588133 0.005670779 0.00480439  0.004039928 0   0   0↙
0.010646289 0.01123963  0.011360531 0.011388431 0.011343791 0.01123963  0.01102573  0.01085275↙
0.010718829 0.010227788 0.010034348 0.009548658 0.009240126 0.008843946 0.005331866 0.001957565↙
0.001186136 0.000718709 0.000435483 0.009763718 0.010065503 0.010091078 0.010046903 0.009961343↙
0.009831143 0.009605617 0.009436822 0.009310342 0.00869282  0.008356928 0.008006014 0.007669818↙
0.001254455 0.000748789 0.000446954 0.000266789 0.013721433 0.014074369 0.014123194 0.014102269↙
0.014035309 0.013923709 0.013566588 0.013449408 0.013030907 0.012869087 0.012549687 0.01221483↙
0.011892585 0.000900184 0.000436854 0.000212003 0.000102884 1.90411E-05 0.007187798 0.007228718↙
0.007172918 0.007070618 0.006955298 0.006810218 0.006430777 0.006317317 0.005789076 0.00551253↙
0.005229214 0.004961374 0.000574741 0   0   0.009706057 0.010012958 0.010078058 0.010096658↙
0.010078058 0.010029698 0.009843698 0.009778598 0.009538657 0.0092543   0.00905196  0.008857896↙
0.001638369 0.000986789 0.000594343 0.000357973 0.010386076 0.01071855  0.010758076 0.01072785↙
```

```
0.01065345  0.01053441  0.010322835 0.010162409 0.009446308 0.009120212 0.008779032 0.008451206↙
0.000800971 0.00035954  0.00016139  7.24447E-05 0.010082522 0.010394538 0.010424763 0.010385238↙
0.010303398 0.010176917 0.009790037 0.009665416 0.009055335 0.008722543 0.008375039 0.008041633↙
0.002343016 0   0.009265981 0.009445006 0.009414781 0.009314806 0.009184605 0.009009765↙
0.008526164 0.008377364 0.007670562 0.007294246 0.006905696 0.00653596  0.000510563 0.000135167↙
6.95478E-05 0.009419617 0.009409387 0.009330337 0.009204786 0.009070866 0.008907186 0.008490545↙
0.008367785 0.007802344 0.007509058 0.007209932 0.006928142 0.001532225 0.000979925 0.000626705↙
0.000400805 0.000256333 0.009973712 0.010556823 0.010673073 0.010696323 0.010647963 0.010540083↙
0.010321533 0.010145762 0.010009982 0.009513361 0.009318061 0.008928142 0.008517329 0.008118358↙
0.000379347 0.000187439 9.26151E-05 1.78759E-05];
    m_exp = [0.664455133    0.660600195 0.659903584 0.653815611 0.643863879 0.643420578 0.6420786↙
0.641032095 0.63    0.6320329   0.630065371 0.6241699   0.61863233  0.527927299 0.511768978↙
0.498364331 0.689057396 0.692621509 0.692304866 0.689900398 0.68720811  0.68546886  0.685555↙
0.684666754 0.687869965 0.680615113 0.688518593 0.669690715 0.667162285 0.66521013  0.643249142↙
0.562980759 0.533264801 0.512741854 0.505587609 0.672333745 0.680039601 0.667376696 0.668113545↙
0.669887703 0.665519991 0.66042195  0.663392543 0.65819495  0.651606294 0.642900574 0.641887469↙
0.635561537 0.536369509 0.520796145 0.51029797  0.492728249 0.71303761  0.6994871   0.701769188↙
0.69999603  0.701257595 0.698408593 0.699423999 0.698485672 0.699962583 0.699467845 0.686372017↙
0.680490058 0.670616312 0.54996579  0.516916765 0.498031168 0.493316219 0.486415845 0.646852772↙
0.641527395 0.6402977   0.642550577 0.641688346 0.640829917 0.634794885 0.629447227 0.622278177↙
0.619749266 0.613333547 0.602327828 0.541584269 0.527157593 0.518919719 0.675408624 0.668449729↙
0.665198697 0.666624644 0.661579087 0.658701847 0.659627084 0.658785575 0.65831585  0.648579795↙
0.642704076 0.639065328 0.568488481 0.537996416 0.522383791 0.512449143 0.698873255 0.692935765↙
0.690407814 0.691673269 0.688984876 0.692774128 0.694173317 0.688442735 0.678472635 0.669053441↙
0.661023285 0.658290052 0.586277093 0.526014283 0.515202825 0.496022157 0.679814469 0.678385666↙
0.678443498 0.675245682 0.675144517 0.678017961 0.670001147 0.671366624 0.669771592 0.659192099↙
0.656571262 0.650316108 0.593335238 0.547726996 0.673297245 0.672809622 0.669223792 0.668933217↙
0.667955017 0.66403757  0.662242917 0.659374749 0.657487586 0.654273822 0.636189927 0.625708532↙
0.500466487 0.473828363 0.457995038 0.693084074 0.690318652 0.687879622 0.68319826  0.684136001↙
0.679103836 0.679561568 0.67289133  0.6701  0.66521 0.65222 0.6294  0.575210798 0.539712115↙
0.529919918 0.527618377 0.515450591 0.686706115 0.679323206 0.684293066 0.689236851 0.684060921↙
0.686483024 0.681287794 0.677176137 0.672125212 0.668292035 0.667041945 0.665278969 0.661918419↙
0.65875516  0.562243403 0.544177804 0.532056223 0.511946383];

    %% Fourier Number

    Fo = a_p*t/R01^2;

    %% Plot Fractional Coverage vs Fourier Number
    figure

    semilogx(Fo, m, Fo_exp, m_exp,'rs')

    xlabel('Fo')

    ylabel('m')

    title('Polystyrene Ellis Model')

    legend('Simulated Data','Experimental Data')

    axis([3E-5 3E-2 0.4 0.9])

    fprintf('--End--\n')
%**************************************************************************
%
```

```
%                               CODE END
%
%*************************************************************************
```

13 Appendix F: MATLAB code for PS: Power Law Model


```
%*************************************************************************
%
%   PROJECT:          GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME:     Power Law Model Simulation vs Experimental Data
%   PURPOSE:          Power Law Simulation for PS
%   DATE:             11/15/2017
%
%*************************************************************************
% clear variables
close all
clc
%% Load Excel Files
% piston speed data references
% % data_points = [2 37 18 9 45 26 50 33 17 53 36];

% delay time
% % data_points = [54 37 72 61 45 79 67 50 84 70 53];

%gas pressure
% % data_points = [87 37 103 94 45 110 99 50 115 102 53 118];

% shot size
% % data_points = [119 37 133 126 45 140 131 50 145 53 147];

%melt temperature
data_points = [148 37 165 155 45 173 160 50 163 53 181];

data_points = data_points-1;

Ti_dp = xlsread('PS_Fractional_Cov_Comp.xlsx','I2:I182'); %degC
m_exp_xls = xlsread('PS_Fractional_Cov_Comp.xlsx','J2:J182');

for data_point_no = 1:length(data_points)
    %find files
    convertedrestime = xlsread('PS_Fractional_Cov_Comp.xlsx','F2:F182');
    convertedrowlength = xlsread('PS_Fractional_Cov_Comp.xlsx','G2:G182');

    dp = data_points(data_point_no);

    converted_res_time=zeros(length(data_points),1);
    converted_rowlength=zeros(length(data_points),1);
    %         factor=zeros(length(data_points),1);

    for i = 1:length(data_points)
        dp = data_points(i);
        converted_res_time(i) = convertedrestime(dp);
        converted_rowlength(i) = convertedrowlength(dp);
        %             factor(i) = factorr(dp);
    end
    %% Initial Conditions

    % Number of nodes
    nodes = 21;

    % time measurements
    t = linspace(1E-10,50,1000); %sec
```

```
% time increments
dt = 0.0001;

% Temperature of mold
Tb = (180-32)/1.8; %degC

% Temperature of melt
Ti = (623.6-32)/1.8; %degC

% Temperature of cooling water
Tc = 25; %degC

%Fractional coverage asymptotic value
n_p = 0.2968;
if n_p >= 0.3
    m_n = 0.1516*log(n_p)+0.7259;
else
    m_n = -0.1428*n_p^2+0.2626*n_p+0.4782;
end


%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m

% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s
```

```
    % Step change in tube
    dr2 = (R02 - R01) / (nodes - 1);

    %% Physical Properties of the polymer

    % Specific Heat
    Cp_p = 2100; %J/(kg*K)

    % Thermal Conductivity
    k_p = 0.15; %W/(m*K)

    % Density
    den_p = 940; %kg/m^3

    % Thermal Diffusivity
    a_p = k_p / (den_p * Cp_p); %m^2/s

    % Flow Activation Energy
    %delta_H_R = 8575; %K
    delta_H_R = 9.0679E3; %K

    % Zero shear viscosity
    zero_visc = 1*10^16*exp(-0.056*(Ti+273.13));

    % Frequency factor
    A = zero_visc / exp(delta_H_R / (25+273.13)); %kg/(m*s)

    % Pressure gradient
    delta_p = 2.42 * 10^6; %Pa

    % Step change in polymer
    dr1 = R01 / (nodes - 1);

    n_dr1 = 1.0 / (nodes - 1);

    %% Physical Properties of the cooling fluid

    % Specific Heat
    Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)

    % Viscosity
    u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/↙
(m*s)

    % Density
    den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

    % Thermal Conductivity
    k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

    % Coefficient of Volume Expansion
    b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

    % Thermal Diffusivity
    a_f = k_f / (den_f * Cp_f); %m^2/s
```

```
%% Heat Transfer outside the tube wall

% Initializing temperature profiles
Temp_p = zeros(nodes,2);
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Tc
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;


% Reynolds Number
Re = 1;

% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);

% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
```

```
            count = count + dt;

            % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
            i=nodes;
            j=1;
            Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+↙
(dr2/2))/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));

            % Temperature profile in Solid
            for i = nodes-1 : -1 : 2

                Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)↙
*Temp_s(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

            end

            % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
            i = 1;
            Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

            Temp_p(nodes,j+1) = Temp_s(i,j+1);

            % Temperature profile in Polymer
            for i = nodes-1 : -1 : 2

                Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p↙
(i)*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)↙
*Temp_p(i+1,j));

            end

            % Boundary Condition 5: @r=0, kdT/dr = 0
            i=1;
            Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));

            for i = 1 : nodes

                Temp_s(i,j) = Temp_s(i,j+1);
                Temp_p(i,j) = Temp_p(i,j+1);

            end

        end

        % Creating Temperature Profiles for Polymer and Solid
        for i = 1 : nodes

            Temp_fp(i,k) = Temp_p(i,j);
            Temp_fs(i,k) = Temp_s(i,j);

        end
    end
```

```
    %% Velocity Profile Calculation

    %initialize velocity matrix
    vel = zeros(nodes,length(t));

    %velocity profile calculation
    for k = 1 : length(t)

        %equation 4.33
        vel(nodes,k) = 0;

        for i = nodes-1 : -1 : 1

            f(i) = (n_dr1/2)*delta_p/(2*L*A)*((r_p(i+1)/exp(delta_H_R/(Temp_fp(i+1,k)+273.13)))+↙
(r_p(i)/exp(delta_H_R/(Temp_fp(i,k)+273.13))));
            vel(i,k) = vel(i+1,k) + f(i);

        end

    end

    %normalizing the velocity profile
    n_vel = zeros(nodes,length(t));

    for k = 1 : length(t)

        for i = 1 : nodes

            n_vel(i,k) = vel(i,k) / vel(1,k);

        end

    end

    %normalizing the radial position
    n_radius = zeros(2*nodes,1);

    for i = 1 : nodes
        n_radius(i) = r_p(i) / R01;
    end

    for i = 1 : nodes
        n_radius(i+nodes) = r_s(i) / R01;
    end

    %% Fractional Coverage Calculation

    %initialize vectors
    m = zeros(length(t),1);
    Rx = zeros(length(t),1);

    for k = 1 : length(t)

        %initializing summation variables
        sum = 0;
```

```
        for i = 1 : nodes-1

            %integrate through ru*
            sum = sum + (n_dr1 / 2) * (n_radius(i)*n_vel(i,k) + n_radius(i+1)*n_vel(i+1,k));

        end

        %integrate through ru*
        Rx(k) = R01 * sqrt(4 * sum);

        %Fractional coverage when integrating through ru*
        m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2-0.02;

    end


    %% Experimental Data

    Fo_exp = [0.009290161   0.009361771 0.009319921 0.00923157  0.00912741  0.00878424↙
0.008636369 0.008528489 0.007097774 0.006588133 0.005670779 0.00480439  0.004039928 0   0   0↙
0.010646289 0.01123963  0.011360531 0.011388431 0.011343791 0.01123963  0.01102573  0.01085275↙
0.010718829 0.010227788 0.010034348 0.009548658 0.009240126 0.008843946 0.005331866 0.001957565↙
0.001186136 0.000718709 0.000435483 0.009763718 0.010065503 0.010091078 0.010046903 0.009961343↙
0.009831143 0.009605617 0.009436822 0.009310342 0.00869282  0.008356928 0.008006014 0.007669818↙
0.001254455 0.000748789 0.000446954 0.000266789 0.013721433 0.014074369 0.014123194 0.014102269↙
0.014035309 0.013923709 0.013566588 0.013449408 0.013030907 0.012869087 0.012549687 0.01221483↙
0.011892585 0.000900184 0.000436854 0.000212003 0.000102884 1.90411E-05 0.007187798 0.007228718↙
0.007172918 0.007070618 0.006955298 0.006810218 0.006430777 0.006317317 0.005789076 0.00551253↙
0.005229214 0.004961374 0.000574741 0   0   0.009706057 0.010012958 0.010078058 0.010096658↙
0.010078058 0.010029698 0.009843698 0.009778598 0.009538657 0.0092543   0.00905196  0.008857896↙
0.001638369 0.000986789 0.000594343 0.000357973 0.010386076 0.01071855  0.010758076 0.01072785↙
0.01065345  0.01053441  0.010322835 0.010162409 0.009446308 0.009120212 0.008779032 0.008451206↙
0.000800971 0.00035954  0.00016139  7.24447E-05 0.010082522 0.010394538 0.010424763 0.010385238↙
0.010303398 0.010176917 0.009790037 0.009665416 0.009055335 0.008722543 0.008375039 0.008041633↙
0.002343016 0   0.009265981 0.009445006 0.009414781 0.009314806 0.009184605 0.009009765↙
0.008526164 0.008377364 0.007670562 0.007294246 0.006905696 0.00653596  0.000510563 0.000135167↙
6.95478E-05 0.009419617 0.009409387 0.009330337 0.009204786 0.009070866 0.008907186 0.008490545↙
0.008367785 0.007802344 0.007509058 0.007209932 0.006928142 0.001532225 0.000979925 0.000626705↙
0.000400805 0.000256333 0.009973712 0.010556823 0.010673073 0.010696323 0.010647963 0.010540083↙
0.010321533 0.010145762 0.010009982 0.009513361 0.009318061 0.008928142 0.008517329 0.008118358↙
0.000379347 0.000187439 9.26151E-05 1.78759E-05];
    m_exp = [0.664455133    0.660600195 0.659903584 0.653815611 0.643863879 0.643420578 0.6420786↙
0.641032095 0.63    0.6320329   0.630065371 0.6241699   0.61863233  0.527927299 0.511768978↙
0.498364331 0.689057396 0.692621509 0.692304866 0.689900398 0.68720811  0.68546886  0.685555↙
0.684666754 0.687869965 0.680615113 0.688518593 0.669690715 0.667162285 0.66521013  0.643249142↙
0.562980759 0.533264801 0.512741854 0.505587609 0.672333745 0.680039601 0.667376696 0.668113545↙
0.669887703 0.665519991 0.66042195  0.663392543 0.65819495  0.651606294 0.642900574 0.641887469↙
0.635561537 0.536369509 0.520796145 0.51029797  0.492728249 0.71303761  0.6994871   0.701769188↙
0.69999603  0.701257595 0.698408593 0.699423999 0.698485672 0.699962583 0.699467845 0.686372017↙
0.680490058 0.670616312 0.54996579  0.516916765 0.498031168 0.493316219 0.486415845 0.646852772↙
0.641527395 0.6402977   0.642550577 0.641688346 0.640829917 0.634794885 0.629447227 0.622278177↙
0.619749266 0.613333547 0.602327828 0.541584269 0.527157593 0.518919719 0.675408624 0.668449729↙
0.665198697 0.666624644 0.661579087 0.658701847 0.659627084 0.658785575 0.65831585  0.648579795↙
0.642704076 0.639065328 0.568488481 0.537996416 0.522383791 0.512449143 0.698873255 0.692935765↙
0.690407814 0.691673269 0.688984876 0.692774128 0.694173317 0.688442735 0.678472635 0.669053441↙
```

```
0.661023285 0.658290052 0.586277093 0.526014283 0.515202825 0.496022157 0.679814469 0.678385666↙
0.678443498 0.675245682 0.675144517 0.678017961 0.670001147 0.671366624 0.669771592 0.659192099↙
0.656571262 0.650316108 0.593335238 0.547726996 0.673297245 0.672809622 0.669223792 0.668933217↙
0.667955017 0.66403757  0.662242917 0.659374749 0.657487586 0.654273822 0.636189927 0.625708532↙
0.500466487 0.473828363 0.457995038 0.693084074 0.690318652 0.687879622 0.68319826  0.684136001↙
0.679103836 0.679561568 0.67289133  0.6701  0.66521 0.65222 0.6294  0.575210798 0.539712115↙
0.529919918 0.527618377 0.515450591 0.686706115 0.679323206 0.684293066 0.689236851 0.684060921↙
0.686483024 0.681287794 0.677176137 0.672125212 0.668292035 0.667041945 0.665278969 0.661918419↙
0.65875516  0.562243403 0.544177804 0.532056223 0.511946383];

    %% Fourier Number

    Fo = a_p*t/R01^2;

    %% Determination of shear rate
    if round(converted_res_time(data_point_no)) == 0
        shear_rate(data_point_no)=0;
    else
    R_bubble = Rx(round(converted_res_time(data_point_no)));
    Q_exp = xlsread('PS_Fractional_Cov_Comp.xlsx','E2:E182');
    factor = Q_exp(data_points(data_point_no))/(3.14*R_bubble^2*n_vel(1,round(converted_res_time↙
(data_point_no))));
    shear_rate(data_point_no) = (factor*n_vel(1,round(converted_res_time(data_point_no)))-↙
factor*n_vel(2,round(converted_res_time(data_point_no))))/(t(2)-t(1));
    end
    %% Plot Viscosity vs shear rate

    viscosity(data_point_no) = A*shear_rate(data_point_no)^(n_p-1);

    sr_vector = logspace(-5, 10, 1E2);
    visc = 0.0000000004*sr_vector.^(n_p-1);

    figure
    loglog(sr_vector,visc)
    hold on
    loglog(shear_rate,viscosity,'rs')


    %% Plot m_sim vs m_exp
    if round(converted_res_time(data_point_no))==0
        m_sim(data_point_no)=0;
        m_exp_dp(data_point_no)=0;
    else
    figure

    m_exp_dp(data_point_no) = m_exp_xls(data_points(data_point_no));
    m_sim_dp(data_point_no) = m(round(converted_res_time(data_point_no)));
    plot(m_sim_dp,m_exp_dp,'rs')
    xlabel('m_{sim}')
    ylabel('m_{exp}')
    hold on
    line = linspace(0.5,1,100);
    plot(line,line)
    end

end
```

```
%% PLOT RESULTS
% sim vs exp
figure
line = linspace(0.1,1,100);
plot(line,line)
% axis([0.4 1 0.4 1])
xlabel('m_{sim}')
ylabel('m_{exp}')
legend('Perfect Simumlation','High 90mm','Base 90mm','Low 90mm','High 285mm','Base 285mm','Low
285mm','High 445mm','Base 445mm','Low 445mm','Base 490mm','Low 490mm');
title('Varying Shot Size Setting')
axis([0.4 1 0.4 1])
box on
legend boxoff
legend('Location','NorthWest')
marker_point =
['s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x',
's','>','x','s','>','x','s','>','x','s','>','x'];
hold on
for i = 1:length(m_sim_dp)
    plot(m_sim_dp(i),m_exp_dp(i),marker_point(i))
end
%% velocity profile
figure
plot(n_vel(1:21,1),n_radius(1:21))
hold on

plot(n_vel(1:21,200),n_radius(1:21))
plot(n_vel(1:21,400),n_radius(1:21))
plot(n_vel(1:21,600),n_radius(1:21))
plot(n_vel(1:21,800),n_radius(1:21))
plot(n_vel(1:21,1000),n_radius(1:21))
plot(n_vel(1:21,1),-n_radius(1:21))
plot(n_vel(1:21,200),-n_radius(1:21))
plot(n_vel(1:21,400),-n_radius(1:21))
plot(n_vel(1:21,600),-n_radius(1:21))
plot(n_vel(1:21,800),-n_radius(1:21))
plot(n_vel(1:21,1000),-n_radius(1:21))
xlabel('Normalized Velocity')
ylabel('Normalized Radius')
legend('t=0s','t=10s','t=20s','t=30s','t=40s','t=50s')
%*************************************************************************
%
%                                CODE END
%
%*************************************************************************
```

14 Appendix G: MATLAB code for PS Simulation Comparison


```
%*************************************************************************
%
%   PROJECT:          GAS ASSISTED INJECTION MOLDING
%   PROGRAM NAME:     Ellis_Model Simulation vs Experimental Data
%   PURPOSE:          m vs Fo for PS
%   DATE:             11/1/2017
%
%*************************************************************************
% clear variables
close all
clc
%% Import Mathematica Solution
% Loading data from Mathematica solution:
% Note: the Mathematica solution solves for the DIMENSIONLESS velocity

% define search directory
search_directory = 'Mathematica_solutions';

% find files in directory
filess = Find_Files(search_directory);
t = linspace(1E-10,50,1000);

for indexx=4:length(filess)
    files = Extract_Files(filess(indexx), filess);

    filename = strcat(search_directory, filesep, files);
    filename = char(filename);
    fid = fopen(filename);
    filedata = textscan(fid, '%f%f', 'Delimiter', ' ', 'Headerlines', 0);
    fclose(fid);
    if indexx == 4
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 7
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 8
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    elseif indexx == 9
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 5
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
    elseif indexx == 6
        %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
        dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
        dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
        dim_r_p = dim_r_p(2:2:end);
        dim_vel = dim_vel(2:2:end);
```

```
elseif indexx == 10
    %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    dim_r_p = dim_r_p(2:2:end);
    dim_vel = dim_vel(2:2:end);
elseif indexx == 11
    %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    dim_r_p = dim_r_p(2:2:end);
    dim_vel = dim_vel(2:2:end);
elseif indexx == 12
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    dim_r_p = dim_r_p(2:2:end);
    dim_vel = dim_vel(2:2:end);
elseif indexx ==13
    %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    dim_r_p = dim_r_p(2:2:end);
    dim_vel = dim_vel(2:2:end);
elseif indexx ==14
    %shorten vectors TURN ON FOR DATA FILES 4 THROUGH 11
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity
    dim_r_p = dim_r_p(2:2:end);
    dim_vel = dim_vel(2:2:end);
end
factor = round(0.9*length(dim_vel));
dim_vel_plot(1:length(dim_vel),indexx) = dim_vel(1:length(dim_vel),1);
dim_r_p_plot(1:length(dim_r_p),indexx) = dim_r_p(1:length(dim_r_p),1);
%% Initial Conditions

% Number of nodes
nodes = length(dim_vel); %this number varies based on the Mathematica solution

% time measurements
t = linspace(1E-10,50,1000); %sec

% time increments
dt = 0.0001;

% Temperature of mold
Tb = (180-32)/1.8; %degC

% Temperature of melt
Ti = (623.6-32)/1.8; %degC

% Temperature of cooling water
Tc = 25; %degC

%Fractional coverage asymptotic value
n_p = 0.2968;
if n_p >= 0.3
```

```
    m_n = 0.1516*log(n_p)+0.7259;
else
    m_n = -0.1428*n_p^2+0.2626*n_p+0.4782;
end

%% Ellis model parameters

% a: (measure of degree of shear thinning behavior)
a = 3.26;

% Tao_half: (value of shear stress at which the apparent
% viscosity has dropped to half its zero shear value)
tao_half = 16618.6;

% Zero Shear Viscosity
% function of temperature
% defined later

%% Physical Properties of the tube

% External Diameter
OD = 2 * (150E-3+0.00635); %m

% Internal Diameter
ID = 2 * 0.00635; %m

% External Radius
R02 = OD * 0.5; %m

% Internal Radius
R01 = ID * 0.5; %m

% Length of tube
L = 0.585; %m

% Gravitational Acceleration
g = 9.81; %m/s^2

% Diameter
D = OD; %m

% Thermal Conductivity
k_s = 16; %W(m*K)

%Density
den_s = 7850; %kg/m^3

%Specific Heat
Cp_s = 514.5; %J/(kg*K)

% Thermal Diffusivity
a_s = k_s / (den_s * Cp_s); %m^2/s

% Step change in tube
dr2 = (R02 - R01) / (nodes - 1);
```

```
    %% Physical Properties of the polymer

    % Specific Heat
    Cp_p = 2100; %J/(kg*K)

    % Thermal Conductivity
    k_p = 0.15; %W/(m*K)

    % Density
    den_p = 940; %kg/m^3

    % Thermal Diffusivity
    a_p = k_p / (den_p * Cp_p); %m^2/s

    % Flow Activation Energy
    %delta_H_R = 8575; %K
    delta_H_R = 9.0679E3; %K

    % Zero shear viscosity
    zero_visc = 1*10^16*exp(-0.056*(Ti+273.13));

    % Frequency factor
    A = zero_visc / exp(delta_H_R / (25+273.13)); %kg/(m*s)

    % Pressure gradient
    delta_p = 2.42 * 10^6; %Pa

    % Step change in polymer
    dr1 = R01 / (nodes - 1);

    n_dr1 = 1.0 / (nodes - 1);

    %% Physical Properties of the cooling fluid

    % Specific Heat
    Cp_f = 10^-6 * Tb^4 - 0.0004 * Tb^3 + 0.0488 * Tb^2 - 2.2527 * Tb + 4212; %J/(kg*K)

    % Viscosity
    u_f = (4 * 10^-8 * Tb^4 - 10^-5 * Tb^3 + 0.001 * Tb^2 - 0.0557 * Tb + 1.7832) * 10^-3; %kg/↙
(m*s)

    % Density
    den_f = -0.0036 * Tb^2 - 0.0656 * Tb + 1000.4; %kg/m^3

    % Thermal Conductivity
    k_f = -9 * 10^-6 * Tb^2 + 0.0021 * Tb + 0.5607; %W/(m*K)

    % Coefficient of Volume Expansion
    b = (0.0036 * 2 * Tb + 0.0656) / den_f; %1/K

    % Thermal Diffusivity
    a_f = k_f / (den_f * Cp_f); %m^2/s

    %% Heat Transfer outside the tube wall

    % Initializing temperature profiles
```

```
Temp_p = zeros(nodes,2);
Temp_s = zeros(nodes,2);
Temp_fp = zeros(nodes,length(t));
Temp_fs = zeros(nodes,length(t));

% Initializing radial position profiles
for i = nodes : -1 : 1
    r_s(i) = R01 + dr2 * (i-1);
end
for i = nodes : -1 : 1
    r_p(i) = dr1 * (i-1);
end

% Boundary Condition 1: @ t=0, T_f(r,0) = Ti
Temp_p = Temp_p + Ti;

% Boundary Condition 2: @ t=0, T_s(r,0) = Tc
Temp_s = Temp_s + Tc;

% Calculation at R01
i=nodes;
j=1;

% Grashof Number
Gr = D^3 * den_f^2 * g * b * (Temp_s(i,j)-Tb) / u_f^2;

% Prandtl Number
Pr = Cp_f * u_f / k_f;

% Raleigh Number
Ra = Gr * Pr;


% Reynolds Number
Re = 1;

% Nusselt Number
Nu = 0.62*Re^(1/2)*Pr^(1/3)/[1+(0.4/Pr)^(2/3)]^(1/4);

% Heat Transfer Coefficient
h = k_f * Nu / D;

%Initialization of counter variables
count = 0;

for k = 1 : length(t)

    disp(length(t)-k+1)

    while count <= t(k)

        %counter variables incrementation
        count = count + dt;

        % Boundary Condition 3: @ r=R02, -ks*dT/dr = h(Ts(R02,t)-Tb)
        i=nodes;
```

```
            j=1;
            Temp_s(i,j+1) = Temp_s(i,j) + (2*a_s*dt/dr2^2)*(Temp_s(i-1,j)-Temp_s(i,j)-((R02+↙
(dr2/2))/R02)*(h/k_s)*dr2*(Temp_s(i,j)-Tb));

            % Temperature profile in Solid
            for i = nodes-1 : -1 : 2

                Temp_s(i,j+1) = Temp_s(i,j) + (a_s*dt/(r_s(i)*dr2^2))*(((r_s(i)+r_s(i-1))/2)↙
*Temp_s(i-1,j)-2*r_s(i)*Temp_s(i,j)+((r_s(i+1)+r_s(i))/2)*Temp_s(i+1,j));

            end

            % Boundary Condition 4: @ r=R01, solid-liquid interface energy balance
            i = 1;
            Temp_s(i,j+1) = Temp_s(i,j) + (2*k_s*dt*(R01+dr2/2)*(Temp_s(i+1,j)-Temp_s(i,j))/(dr2*↙
((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 * Temp_p(nodes,j) + 903)*Cp_p*dr1)))+(2*k_p*dt*↙
(R01-dr1/2)*(Temp_p(nodes-i,j)-Temp_s(i,j))/(dr1*((R01+dr2/4)*den_s*Cp_s*dr2+(R01-dr1/4)*(-0.58 *↙
Temp_p(nodes,j) + 903)*Cp_p*dr1)));

            Temp_p(nodes,j+1) = Temp_s(i,j+1);

            % Temperature profile in Polymer
            for i = nodes-1 : -1 : 2

                Temp_p(i,j+1) = Temp_p(i,j) + ((k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/(r_p↙
(i)*dr1^2))*(((r_p(i)+r_p(i-1))/2)*Temp_p(i-1,j)-2*r_p(i)*Temp_p(i,j)+((r_p(i+1)+r_p(i))/2)↙
*Temp_p(i+1,j));

            end

            % Boundary Condition 5: @r=0, kdT/dr = 0
            i=1;
            Temp_p(i,j+1) = Temp_p(i,j) + (4*(k_p/((-0.58 * Temp_p(i,j) + 903)*Cp_p))*dt/dr1^2)*↙
(Temp_p(i+1,j)-Temp_p(i,j));

            for i = 1 : nodes

                Temp_s(i,j) = Temp_s(i,j+1);
                Temp_p(i,j) = Temp_p(i,j+1);

            end

        end

        % Creating Temperature Profiles for Polymer and Solid
        for i = 1 : nodes

            Temp_fp(i,k) = Temp_p(i,j);
            Temp_fs(i,k) = Temp_s(i,j);

        end
    end

    %% Temperature Profile Calculation
    % figure
    % plot(r_p,Temp_fp(1:nodes,4))
```

```
% hold on
% plot(r_p,Temp_fp(1:nodes,5))
% plot(r_p,Temp_fp(1:nodes,7))

%% Velocity Profile Calculation

%initialize eta_o
eta_o = zeros(nodes,length(t));
%calculate eta_o
for k = 1 : length(t)
    for i = 1 : nodes
        eta_o(i,k) = 1*10^16*exp(-0.056*(Temp_fp(i,k)+273.13));
    end
end

% THE INPUT OF THE MATHEMATHICA SOLUTION IS IMPORTANT FOR THE FOLLOWING
% PORTION OF CODE

%plot dimensionless velocity profile
figure
plot(dim_vel, dim_r_p*0.00635);
hold on
plot(dim_vel, -dim_r_p*0.00635)
ylabel('Radius')
xlabel('Dimensionless Velocity')
title('Dimensionless Velocity Profile for Ellis Model for PS')
axis([0 1 -0.00635 0.00635])

% solving for normalized velocity... eventually
% but first, non-normalized velocity,
% initialize velocity matrix
vel = zeros(nodes,length(t));
% velocity profile calculation
for k = 1 : length(t)
    for i = nodes-1 : -1 : 1

        vel(i,k) = dim_vel(i)*delta_p*R01^2/(L*eta_o(i,k));

    end

end

% Plot Velocity Profile
figure
hold on
plot(vel(1:nodes,1),dim_r_p)
plot(vel(1:nodes,10),dim_r_p)
plot(vel(1:nodes,20),dim_r_p)
plot(vel(1:nodes,30),dim_r_p)
plot(vel(1:nodes,40),dim_r_p)
plot(vel(1:nodes,50),dim_r_p)
plot(vel(1:nodes,60),dim_r_p)
plot(vel(1:nodes,70),dim_r_p)
plot(vel(1:nodes,80),dim_r_p)
plot(vel(1:nodes,90),dim_r_p)
plot(vel(1:nodes,100),dim_r_p)
```

```
plot(vel(1:nodes,1),-dim_r_p)
plot(vel(1:nodes,10),-dim_r_p)
plot(vel(1:nodes,20),-dim_r_p)
plot(vel(1:nodes,30),-dim_r_p)
plot(vel(1:nodes,40),-dim_r_p)
plot(vel(1:nodes,50),-dim_r_p)
plot(vel(1:nodes,60),-dim_r_p)
plot(vel(1:nodes,70),-dim_r_p)
plot(vel(1:nodes,80),-dim_r_p)
plot(vel(1:nodes,90),-dim_r_p)
plot(vel(1:nodes,100),-dim_r_p)
ylabel(' Normalized Radius')
xlabel('Velocity')
title('Velocity Profile for CE Model for PS')

% Now that we can calculate vel_max we can find normaliized velocity!
% normalizing the velocity profile
n_vel = zeros(nodes,length(t));

for k = 1 : length(t)

    for i = 1 : nodes

        n_vel(i,k) = vel(i,k) / vel(1,k);

    end

end

% Plot Normalized Velocity Profile
figure
hold on
plot(n_vel(1:nodes,10),dim_r_p)
%plot(n_vel(1:157,10),dim_r_p)
plot(n_vel(1:nodes,200),dim_r_p)
%plot(n_vel(1:157,30),dim_r_p)
plot(n_vel(1:nodes,400),dim_r_p)
%plot(n_vel(1:157,50),dim_r_p)
plot(n_vel(1:nodes,600),dim_r_p)
%plot(n_vel(1:157,70),dim_r_p)
plot(n_vel(1:nodes,800),dim_r_p)
%plot(n_vel(1:157,90),dim_r_p)
plot(n_vel(1:nodes,1000),dim_r_p)
plot(n_vel(1:nodes,10),-dim_r_p)
%plot(n_vel(1:157,10),-dim_r_p)
plot(n_vel(1:nodes,200),-dim_r_p)
%plot(n_vel(1:157,30),-dim_r_p)
plot(n_vel(1:nodes,400),-dim_r_p)
%plot(n_vel(1:157,50),-dim_r_p)
plot(n_vel(1:nodes,600),-dim_r_p)
%plot(n_vel(1:157,70),-dim_r_p)
plot(n_vel(1:nodes,800),-dim_r_p)
%plot(n_vel(1:157,90),-dim_r_p)
plot(n_vel(1:nodes,100),-dim_r_p)
ylabel('Normalized Radius')
xlabel('Normalized Velocity')
```

```
    title('Normalized Velocity Profile for Ellis Model for PS')
    box on
    legend('t=1 sec','t=20 sec','t=40 sec','t=60 sec','t=80 sec','t=100 sec')

    %%

    for g = 1:1000
        clear r_p
        for i = nodes : -1 : 1
        r_p(i) = dr1 * (i-1);
        end
        Qm=trapz(r_p,vel(1:nodes,g));
       Q_m(indexx,g)=Qm;
    end
end
%% Plot All Dimensionless Velocity Profiles
figure
plot(dim_vel_plot(1:173,4),dim_r_p_plot(1:173,4))
hold on
plot(dim_vel_plot(1:173,4),-dim_r_p_plot(1:173,4))
% title('\tau_{wall}=1.47E6 Pa')
ylabel('dimensionless radius')
xlabel('dimensionless velocity')
for i = 5:length(filess)
    plot(dim_vel_plot(1:173,i),dim_r_p_plot(1:173,i))
end
for i = 5:length(filess)
    plot(dim_vel_plot(1:173,i),-dim_r_p_plot(1:173,i))
end
title('All Dimensionless Velocity Profiles')

%% Fractional Coverage Calculation

    %initialize vectors
    m = zeros(length(t),1);
    Rx = zeros(length(t),1);

    %define shear stress average and wall
    tau_avg = linspace(1.47E6,3.42,length(t));
    tau_wall = tau_avg(1); %isothermal

    % Rx calculation
    for k = 1 : length(t)

        Rx(k) = R01*sqrt(tau_avg(k)*eta_o(1,k)/(tau_wall*eta_o(105,k)));

        %Fractional coverage when integrating through ru*
        m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2;

    end


    %% Experimental Data

    Fo_exp = [0.009290161   0.009361771 0.009319921 0.00923157  0.00912741  0.00878424↙
0.008636369 0.008528489 0.007097774 0.006588133 0.005670779 0.00480439  0.004039928 0   0   0↙
```

```
0.010646289 0.01123963  0.011360531 0.011388431 0.011343791 0.01123963  0.01102573  0.01085275↙
0.010718829 0.010227788 0.010034348 0.009548658 0.009240126 0.008843946 0.005331866 0.001957565↙
0.001186136 0.000718709 0.000435483 0.009763718 0.010065503 0.010091078 0.010046903 0.009961343↙
0.009831143 0.009605617 0.009436822 0.009310342 0.00869282  0.008356928 0.008006014 0.007669818↙
0.001254455 0.000748789 0.000446954 0.000266789 0.013721433 0.014074369 0.014123194 0.014102269↙
0.014035309 0.013923709 0.013566588 0.013449408 0.013030907 0.012869087 0.012549687 0.01221483↙
0.011892585 0.000900184 0.000436854 0.000212003 0.000102884 1.90411E-05 0.007187798 0.007228718↙
0.007172918 0.007070618 0.006955298 0.006810218 0.006430777 0.006317317 0.005789076 0.00551253↙
0.005229214 0.004961374 0.000574741 0   0   0.009706057 0.010012958 0.010078058 0.010096658↙
0.010078058 0.010029698 0.009843698 0.009778598 0.009538657 0.0092543   0.00905196  0.008857896↙
0.001638369 0.000986789 0.000594343 0.000357973 0.010386076 0.01071855  0.010758076 0.01072785↙
0.01065345  0.01053441  0.010322835 0.010162409 0.009446308 0.009120212 0.008779032 0.008451206↙
0.000800971 0.00035954  0.00016139  7.24447E-05 0.010082522 0.010394538 0.010424763 0.010385238↙
0.010303398 0.010176917 0.009790037 0.009665416 0.009055335 0.008722543 0.008375039 0.008041633↙
0.002343016 0   0.009265981 0.009445006 0.009414781 0.009314806 0.009184605 0.009009765↙
0.008526164 0.008377364 0.007670562 0.007294246 0.006905696 0.00653596  0.000510563 0.000135167↙
6.95478E-05 0.009419617 0.009409387 0.009330337 0.009204786 0.009070866 0.008907186 0.008490545↙
0.008367785 0.007802344 0.007509058 0.007209932 0.006928142 0.001532225 0.000979925 0.000626705↙
0.000400805 0.000256333 0.009973712 0.010556823 0.010673073 0.010696323 0.010647963 0.010540083↙
0.010321533 0.010145762 0.010009982 0.009513361 0.009318061 0.008928142 0.008517329 0.008118358↙
0.000379347 0.000187439 9.26151E-05 1.78759E-05];
    m_exp = [0.664455133    0.660600195 0.659903584 0.653815611 0.643863879 0.643420578 0.6420786↙
0.641032095 0.63    0.6320329   0.630065371 0.6241699   0.61863233  0.527927299 0.511768978↙
0.498364331 0.689057396 0.692621509 0.692304866 0.689900398 0.68720811  0.68546886  0.685555↙
0.684666754 0.687869965 0.680615113 0.688518593 0.669690715 0.667162285 0.66521013  0.643249142↙
0.562980759 0.533264801 0.512741854 0.505587609 0.672333745 0.680039601 0.667376696 0.668113545↙
0.669887703 0.665519991 0.66042195  0.663392543 0.65819495  0.651606294 0.642900574 0.641887469↙
0.635561537 0.536369509 0.520796145 0.51029797  0.492728249 0.71303761  0.6994871   0.701769188↙
0.69999603  0.701257595 0.698408593 0.699423999 0.698485672 0.699962583 0.699467845 0.686372017↙
0.680490058 0.670616312 0.54996579  0.516916765 0.498031168 0.493316219 0.486415845 0.646852772↙
0.641527395 0.6402977   0.642550577 0.641688346 0.640829917 0.634794885 0.629447227 0.622278177↙
0.619749266 0.613333547 0.602327828 0.541584269 0.527157593 0.518919719 0.675408624 0.668449729↙
0.665198697 0.666624644 0.661579087 0.658701847 0.659627084 0.658785575 0.65831585  0.648579795↙
0.642704076 0.639065328 0.568488481 0.537996416 0.522383791 0.512449143 0.698873255 0.692935765↙
0.690407814 0.691673269 0.688984876 0.692774128 0.694173317 0.688442735 0.678472635 0.669053441↙
0.661023285 0.658290052 0.586277093 0.526014283 0.515202825 0.496022157 0.679814469 0.678385666↙
0.678443498 0.675245682 0.675144517 0.678017961 0.670001147 0.671366624 0.669771592 0.659192099↙
0.656571262 0.650316108 0.593335238 0.547726996 0.673297245 0.672809622 0.669223792 0.668933217↙
0.667955017 0.66403757  0.662242917 0.659374749 0.657487586 0.654273822 0.636189927 0.625708532↙
0.500466487 0.473828363 0.457995038 0.693084074 0.690318652 0.687879622 0.68319826  0.684136001↙
0.679103836 0.679561568 0.67289133  0.6701  0.66521 0.65222 0.6294  0.575210798 0.539712115↙
0.529919918 0.527618377 0.515450591 0.686706115 0.679323206 0.684293066 0.689236851 0.684060921↙
0.686483024 0.681287794 0.677176137 0.672125212 0.668292035 0.667041945 0.665278969 0.661918419↙
0.65875516  0.562243403 0.544177804 0.532056223 0.511946383];

    %% Fourier Number

    Fo = a_p*t/R01^2;

    %% Plot Fractional Coverage vs Fourier Number
    figure

    semilogx(Fo, m, Fo_exp, m_exp,'rs')

    xlabel('Fo')
```

```
    ylabel('m')

    title('Polystyrene Ellis Model')

    legend('Simulated Data','Experimental Data')

    axis([3E-5 3E-2 0.4 0.9])

%%
close all
%% Determine simulated vs experimental data: determination of \tau_{wall}
    dp=linspace(1,181,181);
    for dp=1:length(dp)
        disp(dp)
convertedrestime = xlsread('PS_Fractional_Cov_Comp.xlsx','F2:F182');
convertedrowlength = xlsread('PS_Fractional_Cov_Comp.xlsx','G2:G182');
Q_exp = xlsread('PS_Fractional_Cov_Comp.xlsx','E2:E182');
for i = 1:length(convertedrestime)
convertedrestime(i) = round(convertedrestime(i));
convertedrowlength(i) = round(convertedrowlength(i));
end
if convertedrestime(dp) == 0
    disp('Error')
else
tau = linspace(1.168E5,1.47E6,8);
for i = 1:8
    Q_e(i)=Q_exp(dp);
end
Q_m_ref=Q_m(7:length(filess),convertedrestime(dp));
[tau_wall_int(dp),Q_e_int(dp)]=intersections(tau,Q_m_ref,tau,Q_e);
% figure
% loglog(tau,Q_m_ref)
% hold on
% loglog(tau,Q_e)
% xlabel('\tau_{wall} (Pa-s^2)')
% ylabel('Q (m/s)')
end
    end

%% Determine simulated vs experimental data: solve for Rx
%tau_wall_int range = [3E5, 6.6E5]
% Based on solved dimensionless velocity profiles estimate tau_wal~3E5
indexx=13;
if indexx == 13
    disp('i')
    files = Extract_Files(filess(indexx), filess);

    filename = strcat(search_directory, filesep, files);
    filename = char(filename);
    fid = fopen(filename);
    filedata = textscan(fid, '%f%f', 'Delimiter', ' ', 'Headerlines', 0);
    fclose(fid);
    dim_r_p = cell2mat(filedata(:,1)); %dimensionless r_p
    dim_vel = -cell2mat(filedata(:,2)); %dimensionless velocity

    %initialize eta_o
```

```
    eta_o = zeros(nodes,length(t));
    %calculate eta_o
    for k = 1 : length(t)
        for i = 1 : nodes
            eta_o(i,k) = 1*10^16*exp(-0.056*(Temp_fp(i,k)+273.13));
        end
    end

    % initialize velocity matrix
    vel = zeros(nodes,length(t));
    % velocity profile calculation
    for k = 1 : length(t)
        for i = nodes-1 : -1 : 1

            vel(i,k) = dim_vel(i)*delta_p*R01^2/(L*eta_o(i,k));

        end

    end

    n_vel = zeros(nodes,length(t));

    for k = 1 : length(t)

        for i = 1 : nodes

            n_vel(i,k) = vel(i,k) / vel(1,k);

        end

    end
% Fractional Coverage Calculation

    %initialize vectors
    m = zeros(length(t),1);
    Rx = zeros(length(t),1);

    %define shear stress average and wall
    tau_avg = linspace(1.47E6,3.42,length(t));
    tau_wall = tau_avg(1); %isothermal

    % Rx calculation
    for k = 1 : length(t)

        Rx(k) = R01*sqrt(tau_avg(k)*eta_o(1,k)/(tau_wall*eta_o(105,k)));

        %Fractional coverage when integrating through ru*
        m(k) = 1 - (1 - m_n) * (Rx(k) / R01)^2;

    end
end
%% Determine simulated vs experimental data: determination of viscosity

% determine m_sim
dp=linspace(1,181,181);
for i=1:length(dp)
```

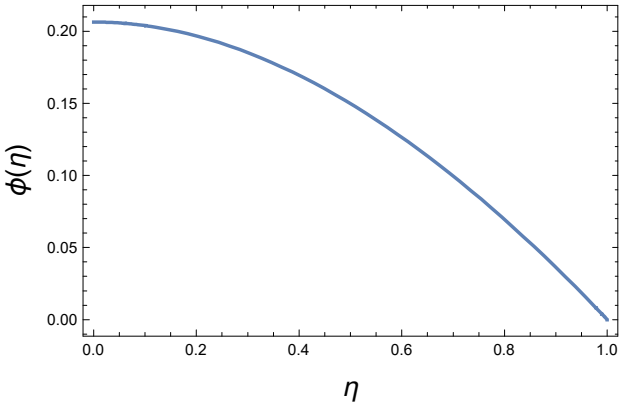
```
    m_sim = zeros(length(convertedrestime),1);

    for i = 1:length(convertedrestime)
        if convertedrestime(i) == 0
            disp('Error')

        elseif convertedrestime(i) < 1001
            m_sim(i) = m(convertedrestime(i));
        else
            m_sim(i) = 0;
        end
    end
end

% calculate vel_sim
vel_sim = zeros(length(convertedrowlength),1);
for i = 1:length(convertedrowlength)
    if convertedrestime(i) == 0
        disp('Error')
    else
        vel_sim(i,i) = vel(convertedrowlength(i),convertedrestime(i));
    end
end

% calculate shear rate (gamma_dot)
vel_diff = zeros(length(vel_sim),1);
t_diff = zeros(length(vel_sim),1);
gamma_dot = zeros(length(vel_sim),1);
for i =1:length(vel_sim)
    if convertedrestime(i) == 0
        disp('Error')
    else
        vel_diff(i) = vel(convertedrowlength(i)-1,convertedrestime(i))-vel(convertedrowlength(i),↙
convertedrestime(i));
        t_diff(i) = 0.05;
        gamma_dot(i) = vel_diff(i)/t_diff(i);
    end
end
%%
% piston speed data references
% data_points = [2 37 18 9 45 26 50 33 53 36];

% delay time
% data_points = [54 37 72 61 45 79 67 50 84 70 53];

%gas pressure
% data_points = [87 37 103 94 45 110 99 50 115 102 53 118];

% shot size
data_points = [119 37 133 126 45 140 131 50 145 53 147];

%melt temperature
% data_points = [148 37 165 155 45 173 160 50 163 53 181];

Ti_dp = xlsread('PS_Fractional_Cov_Comp.xlsx','I2:I182'); %degC
```

```
for data_point_no = 1:length(data_points)
    dp = data_points(data_point_no);
% plot viscosity
lenn = 1E2;
Ti = Ti+273.15; %K
m_power = A.*exp(delta_H_R./Ti); %Pa-s
shear_rate = logspace(-3, 10, lenn);
n_p = 0.2968;
B = 1.6783E-6;
Tb = 1.1376E4;
no = B.*exp(Tb./Ti_dp(data_points(data_point_no)));
no_exp = B*exp(Tb/Ti_dp(data_points(data_point_no)));
tau_star = 1.738E4;
for k = 1:length(shear_rate)
    for i = 1:length(Ti)
        viscosity(i,k) = no(i)/(1+(no(i)*shear_rate(k)/tau_star)^(1-n_p));
        visc_dp(data_point_no,k) = viscosity(i,k);
    end
end
figure (3)
loglog(shear_rate, viscosity(1,1:lenn))
hold on

% plot data points
for i = 1:length(gamma_dot)
    visc_exp(i) = no_exp/(1+(no_exp*gamma_dot(i)/tau_star)^(1-n_p));
end
visc_exp_dp(data_point_no) = visc_exp(data_point_no);
loglog(gamma_dot(1), visc_exp(1), 'rs')
leg_ent = ['ABCDEFGHIJKLMN'];
legend(leg_ent(1))
xlabel('\gamma_{dot} [1/s]')
ylabel('\eta [Pa-s]')
hold off

% Plot m_sim vs m_exp
figure (4)
m_exp_dp(data_point_no) = [m_exp(data_points(data_point_no)-1)];
m_sim_dp(data_point_no) = m_sim(data_points(data_point_no)-1);
plot(m_sim_dp(data_point_no), m_exp_dp(data_point_no),'rs')
xlabel('m_{sim}')
ylabel('m_{exp}')
leg_ent =↙
['ABCDEFGHIJKLMNABCDEFGHIJKLMNABCDEFGHIJKLMNABCDEFGHIJKLMNABCDEFGHIJKLMNABCDEFGHIJK↙
LMN'];
legend(leg_ent(data_point_no))
axis([0.5 1 0.5 1])
hold on
line = linspace(0.5,1,100);
plot(line,line)
end
%% Define Data Points

visc_exp_dpp=visc_exp_dp;
  clear visc_exp_dp
for i = 1:length(data_points)
```

```
    visc_exp_dp(i)=visc_exp_dpp(i);
end

%% PLOT RESULTS
figure(100)
loglog(shear_rate, visc_dp(1,1:lenn))
marker_point =↙
['s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x',↙
's','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','↙
s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s↙
','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s'↙
,'>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s',↙
'>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x','s','>','x'];
 hold on
 for i = 2:length(data_points)
 loglog(shear_rate, visc_dp(i,1:lenn))
 end
 for i = 1:length(data_points)
 loglog(gamma_dot(i), visc_exp_dp(i), marker_point(i))
 end
title('Varying Shot Size Setting')
legend('High 90mm','Base 90mm','Low 90mm','High 285mm','Base 285mm','Low 285mm','High↙
445mm','Base 445mm','Low 445mm','Base 490mm','Low 490mm','High 90mm','Base 90mm','Low 90mm','High↙
285mm','Base 285mm','Low 285mm','High 445mm','Base 445mm','Low 445mm','Base 490mm','Low 490mm')
legend boxoff
box on
legend('Location','SouthWest')
set(legend,'FontSize',7)
xlabel('\gamma_{dot} [1/s]')
ylabel('\eta [Pa-s]')
 hold off

 figure(200)
 hold on
 for i = 1:length(data_points)
     plot(m_sim_dp(i), m_exp(data_points(i)-1),marker_point(i))
 end
 line = linspace(0.1,1,100);
plot(line,line)
 xlabel('m_{sim}')
ylabel('m_{exp}')
legend('High 90mm','Base 90mm','Low 90mm','High 285mm','Base 285mm','Low 285mm','High↙
445mm','Base 445mm','Low 445mm','Base 490mm','Low 490mm','Perfect Simulation');
title('Varying Shot Size Setting')
axis([0.4 1 0.4 1])
box on
legend boxoff
legend('Location','NorthWest')

%*************************************************************************
%
%                                CODE END
%
%*************************************************************************
```

15 Appendix H: Mathematica code for PS: Cross-Exponential Model

```
ClearAll["Global`*"]
R = 0.00635; (*m*)
delPL = 2420000; (*Pa*)
T = {358.209114772210, 358.209114772210, 358.209114772210, 358.209114772210,
   58.209114780497, 358.209115153781, 358.209121591472, 358.209179879124,
   358.209515036587, 358.210896401797, 358.215331208706, 358.227068418781,
   358.253776265434, 358.307660841144, 358.406235985968, 358.572807581777,
   358.836211202275, 359.230514067271, 359.794243069220, 360.568981457907,
   361.599279718692, 362.930476581638, 364.609240053483, 366.682680892601,
   369.196418805698, 372.197481216343, 375.730237013768, 379.840434898080,
   384.574157656625, 389.974803415770, 396.090373508991, 402.965284687088,
   410.648969074409, 419.193437607716, 428.647966474025, 439.071922380453,
   450.520214711697, 463.057811129546, 476.755034731154, 491.678976970746,
   507.913687369223, 525.536983450300, 544.642942462401, 565.334186087718,
   587.708893295345, 611.891068945741, 637.995652101931, 666.161982544949,
   696.542081489387, 729.281613359925, 764.564208566193, 802.560353591851,
   843.484644461334, 887.536459767833, 934.956602799835, 986.007306519566,
   1040.94028710230, 1100.07114484117, 1163.69356483919, 1232.16115143828,
   1305.85849137154, 1385.15502120982, 1470.51239551765, 1562.36058630588,
   1661.21596287046, 1767.63947149223, 1882.16993489342, 2005.47902327938,
   2138.19219486422, 2281.05914656413, 2434.89328904772, 2600.47514789544,
   2778.77620125508, 2970.69979204574, 3177.32733873120, 3399.83056114398,
   3639.33143219877, 3897.22558063904, 4174.80715214072, 4473.62448406337,
   4795.35262066395, 5141.59027996188, 5514.32682289482, 5915.39927103894,
   6347.00565279730, 6811.51976894279, 7311.19725328897, 7848.84866226162,
   8427.05481562460, 9048.90618660127, 9717.73402951443, 10436.6860165114,
   11209.6935595937, 12040.3412304040, 12932.9282520195, 13892.0792025874,
   14922.1339080227, 16028.5318429920, 17216.1900243504, 18491.0201498321}; (*deg C*)
μ0 = 1.6783 * 10^-6 * Exp[1.1376 * 10^4 / T]; (*NOT CONSTANT*) (*Pa-s*)
vv = (delPL) R^2 / μ0; (*NOT CONSTANT*) (*1/s*)
τ = μ0 * vv / (1.738 * 10^4 * R); (*CONSTANT*) (*dimensionless*)
n = 0.2968; τ = 0.884177; (*BOTH CONSTANT*)
system[Ω_] :=
  {ϕ'[η] == ψ[η], ∂η (η (1 + (τ ψ[η]))^(n-1) ψ[η]) == -η, ϕ[0.0001`] == Ω, ψ[0.0001`] == 0};
myODEsoln[Ω_?NumericQ] := NDSolve[system[Ω], {ψ[η], ϕ[η]}, {η, 0.0001`, 1}];
endpoint[Ω_?NumericQ] := First[ϕ[η] /.myODEsoln[Ω] /.η → 1]
bc = FindRoot[endpoint[Ω], {Ω, -2, 2}];
Plot[Evaluate[ϕ[η] /.myODEsoln[Ω /.bc]], {η, 0.0001`, 1}, PlotStyle → Thick,
 Frame → True, FrameLabel → {Style["η", 16], Style["ϕ(η)", 16]}]
(*Extraction of Data from Plot*)
data = Cases[Plot[Evaluate[ϕ[η] /.myODEsoln[Ω /.bc]], {η, 0.0001`, 1}],
    Line[data_] :> data, -4, 1][[1]];
Export["CE_Model_PS_data.txt", data, "Table"]
```



CE_Model_PS_data.txt

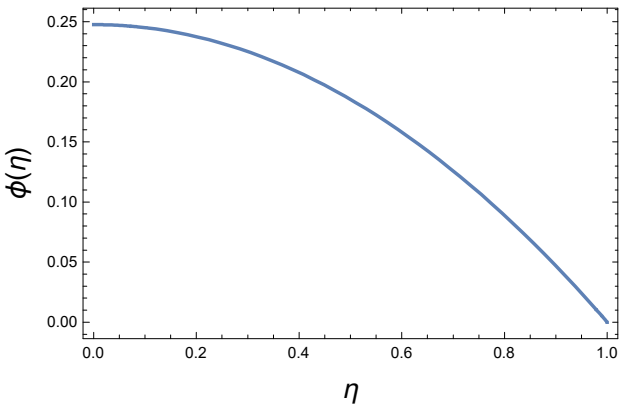
```
ϕ1 = First[ϕ[η] /.myODEsoln[Ω /.bc]];
Q = NIntegrate[2 π η ϕ1, {η, 0.0001`, 1}]
```

```
0.312047
```

```
Rx = 0.0003;
delPL = 2420000;
T = {252};
μ0 = 1.6783 * 10^-6 * Exp[1.1376 * 10^4 / T] (*NOT CONSTANT*)
vv = (delPL) Rx^2 / μ0; (*NOT CONSTANT*)
τ = μ0 * vv / (1.738 * 10^4 * Rx) (*CONSTANT*)
n = 0.2968; τ = 0.04177215189873417; (*BOTH CONSTANT*)
(*BECAUSE BOTH PARAMETERS ARE NOT FUNCTIONS OF TEMPERATURE FOR THIS NORMALIZED SOLUTION,
TEMPERATURE DEPENDENCE WOULD ONLY APPEAR IN PLOTS OF R VS UZ CORRECT? SINCE UZ=
 phi*vv and vv is a function of temperature*)
system[Ω_] := {ϕ'[η] == ψ[η], ∂η (η (1 + (τ ψ[η]))^(n-1) ψ[η]) == -η,
   ϕ[0.0001`] == Ω, ψ[0.0001`] == 0};
myODEsoln[Ω_?NumericQ] := NDSolve[system[Ω], {ψ[η], ϕ[η]}, {η, 0.0001`, 1}];
endpoint[Ω_?NumericQ] := First[ϕ[η] /.myODEsoln[Ω] /.η → 1]
bc = FindRoot[endpoint[Ω], {Ω, -2, 2}];
Plot[Evaluate[ϕ[η] /.myODEsoln[Ω /.bc]], {η, 0.0001`, 1}, PlotStyle → Thick,
 Frame → True, FrameLabel → {Style["η", 16], Style["ϕ(η)", 16]}]
ϕ1 = First[ϕ[η] /.myODEsoln[Ω /.bc]];
Q = NIntegrate[2 π η ϕ1, {η, 0.0001`, 1}]
```

$\{6.76337 \times 10^{13}\}$

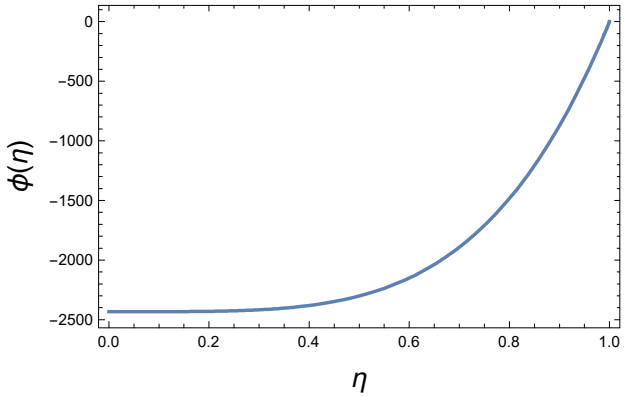
$\{0.0417722\}$



0.388129

16 Appendix I: Mathematica code for PS: Ellis Model

```
ClearAll["Global`*"]
R = 0.00635; (*m*)
delPL = 2420000; (*Pa/m*)
(*NOTE: PARAMETERS ARE FOR PS*)
T = {358.209114772210, 358.209114772210, 358.209114772210, 358.209114772210,
   58.209114780497, 358.209115153781, 358.209121591472, 358.209179879124,
   358.209515036587, 358.210896401797, 358.215331208706, 358.227068418781,
   358.253776265434, 358.307660841144, 358.406235985968, 358.572807581777,
   358.836211202275, 359.230514067271, 359.794243069220, 360.568981457907,
   361.599279718692, 362.930476581638, 364.609240053483, 366.682680892601,
   369.196418805698, 372.197481216343, 375.730237013768, 379.840434898080,
   384.574157656625, 389.974803415770, 396.090373508991, 402.965284687088,
   410.648969074409, 419.193437607716, 428.647966474025, 439.071922380453,
   450.520214711697, 463.057811129546, 476.755034731154, 491.678976970746,
   507.913687369223, 525.536983450300, 544.642942462401, 565.334186087718,
   587.708893295345, 611.891068945741, 637.995652101931, 666.161982544949,
   696.542081489387, 729.281613359925, 764.564208566193, 802.560353591851,
   843.484644461334, 887.536459767833, 934.956602799835, 986.007306519566,
   1040.94028710230, 1100.07114484117, 1163.69356483919, 1232.16115143828,
   1305.85849137154, 1385.15502120982, 1470.51239551765, 1562.36058630588,
   1661.21596287046, 1767.63947149223, 1882.16993489342, 2005.47902327938,
   2138.19219486422, 2281.05914656413, 2434.89328904772, 2600.47514789544,
   2778.77620125508, 2970.69979204574, 3177.32733873120, 3399.83056114398,
   3639.33143219877, 3897.22558063904, 4174.80715214072, 4473.62448406337,
   4795.35262066395, 5141.59027996188, 5514.32682289482, 5915.39927103894,
   6347.00565279730, 6811.51976894279, 7311.19725328897, 7848.84866226162,
   8427.05481562460, 9048.90618660127, 9717.73402951443, 10436.6860165114,
   11209.6935595937, 12040.3412304040, 12932.9282520195, 13892.0792025874,
   14922.1339080227, 16028.5318429920, 17216.1900243504, 18491.0201498321};
μ0 = 2 * 10^8; (*1.6783*10^-6*Exp[1.1376*10^4/T]*); (*NOT CONSTANT*)
vv = (delPL) R^2 / μ0; (*NOT CONSTANT*)
n = 3.26; τ = 1.329 * 10^6 / 16618.6; (*BOTH CONSTANT*)
system[Ω_] :=
 {ϕ'[η] == ψ[η], ∂η (η ((1 + (τ η))^(n-1))^-1 ψ[η]) == η, ϕ[0.0001`] == Ω, ψ[0.0001`] == 0};
myODEsoln[Ω_?NumericQ] := NDSolve[system[Ω], {ψ[η], ϕ[η]}, {η, 0.0001`, 1}];
endpoint[Ω_?NumericQ] := First[ϕ[η] /.myODEsoln[Ω] /.η → 1]
bc = FindRoot[endpoint[Ω], {Ω, -2, 2}];
Plot[Evaluate[ϕ[η] /.myODEsoln[Ω /.bc]], {η, 0.0001`, 1}, PlotStyle → Thick,
 Frame → True, FrameLabel → {Style["η", 16], Style["ϕ(η)", 16]}]
(*Extraction of Data from Plot*)
data = Cases[Plot[Evaluate[ϕ[η] /.myODEsoln[Ω /.bc]], {η, 0.0001`, 1}],
    Line[data_] :> data, -4, 1][[1]];
Export["Ellis_Model_PS_data_wallstress10.txt", data, "Table"]
```



Ellis_Model_PS_data_wallstress10.txt

17 Appendix J: Data for PC

Fourier.and.Capillary.number.analysis.for.the.polycarbonate.data.from.the.second.set.of.experiments	
Thermal.Diffusivity.(m^2/s)	0.0000001153
n	0.03727
Tb.(K)	9067.9
T.(Pa)	942210
B.(Pa-sec)	0.00010242
R.(m)	0.00635

High.delay.time.setting	Rowlength.(mm)	Residual.Time.(sec)	Fourier.#	Bubble.velocity.(m/sec)	Solid.layer.fractional.coverage	Adjusted.Radius.(m)
	90	6.3404	0.096232248	0.273871583	0.084989024	0.006074169
	145	6.11765	0.09285143	0.356615911	0.087933406	0.006064369
	170	5.9964	0.091011143	0.402083317	0.08887481	0.006061258
	195	5.86265	0.088981135	0.453347674	0.096284459	0.006036561
	215	5.74665	0.087220529	0.499026561	0.094232915	0.006043409
	235	5.62265	0.085338503	0.54910801	0.094327765	0.006043093
	260	5.4564	0.082815222	0.619343052	0.093217665	0.006046794
	275	5.35065	0.081210188	0.665580321	0.093165485	0.006046969
	285	5.27765	0.080102221	0.698307341	0.09214729	0.006050363
	325	4.96565	0.0753668	0.846118417	0.09532733	0.006039757
	343	4.81481	0.073077406	0.922474122	0.096103987	0.006037164
	360	4.6664	#	1.000904453	0.097940442	0.006031028
	375	4.53065	0.068764531	1.224843384	0.098039329	0.006030697
	403	4.26521	0.064735781	2.471832128	0.096075171	0.00603726
	415	4.14665	0.062936321	3.340630722	0.095403502	0.006039503
	422	4.07616	0.061886645	3.982300343	0.095091323	0.006040545
	445	3.2945	0.050002703	7.093363496	0.085676014	0.006071888
	460	2.519	0.038232451	10.33627576	0.074570815	0.006108651
	475	1.7435	0.026462199	15.06176823	0.061847263	0.006150501
	490	0.968	0.014691946	21.94764025	0.04702575	0.006198395
	505	0.1925	0.002921694	31.98156452	0.030134984	0.00625359
	525	.	0	52.83424859	0.02187	0.006280015
Base.setting	90	5.4356	0.082499528	0.273922424	0.074654356	0.006108375
	145	5.181775	0.078647066	0.345103697	0.076602478	0.006101942
	170	5.0484	0.076622749	0.383310338	0.07712481	0.006100216
	195	4.903775	0.074427684	0.4257466	0.082045974	0.00608393
	215	4.779975	0.072548694	0.463054586	0.088662251	0.006085511
	235	4.648975	0.070560424	0.503631547	0.080769336	0.006088159
	260	4.4751	0.067921414	0.559368903	0.080307653	0.006089688
	275	4.365375	0.066256048	0.595764195	0.080463033	0.006089173
	285	4.289975	0.065111655	0.62131919	0.077964185	0.006090825
	315	4.052975	0.061514556	0.704751278	0.08450179	0.006075786
	325	3.970375	0.060260884	0.734981217	0.083660524	0.006078577
	343	3.817159	0.057935428	0.792700882	0.08534883	0.006072975
	360	3.6671	0.055657888	0.851368405	0.087279169	0.006066563
	375	3.530375	0.053582727	1.126278796	0.086603582	0.006068808
	403	3.264319	0.049544627	1.977273938	0.083470457	0.006079207
	415	3.145975	0.047748441	2.516625954	0.08241347	0.006082712
	422	3.075744	0.046682505	2.896835972	0.081719006	0.006085013
	445	2.6165	0.03971227	4.599368895	0.07353448	0.006105547
	460	2.021	0.030673991	6.217818348	0.066601132	0.006134898
	475	1.4255	0.021633712	8.405776073	0.056535686	0.006168473
	490	0.83	0.012597433	11.36364355	0.044664425	0.00620722
Low.delay.time.setting	90	4.61	0.069966876	0.282386729	0.068210023	0.006139609
	145	4.367175	0.066283368	0.35576753	0.070536145	0.006121953
	170	4.2388	0.06433494	0.395154771	0.071423623	0.006119029
	195	4.099175	0.062215763	0.438902596	0.07389319	0.006107459
	215	3.979375	0.060397483	0.477363145	0.074253938	0.006109697
	235	3.852375	0.058469524	0.51919395	0.074492175	0.006108911
	260	3.6835	0.055906801	0.576674229	0.074543862	0.006109139
	275	3.576775	0.054286969	0.614173531	0.074236188	0.00610874
	285	3.503375	0.053172931	0.640518185	0.077478327	0.006109755
	315	3.272375	0.049666898	0.726528355	0.078005671	0.006099047
	343	3.042159	0.046172765	0.81719481	0.0776754	0.006097304
	375	2.761775	0.0419172	0.934748503	0.073802996	0.006098091
	403	2.501319	0.037964095	1.051399605	0.072477973	0.006111185
	415	2.385375	0.03620433	1.105748226	0.071505505	0.006115555
	422	2.316544	0.035159649	1.138719824	0.067532756	0.00611876
	445	2.084175	0.031632838	1.254230495	0.07439832	0.006131836
High.piston.speed.setting	90	5.4086	0.082089732	0.351022374	0.076258668	0.00610922
	145	5.1457	0.078099533	0.437400812	0.076765599	0.006103078
	170	5.0102	0.076042964	0.483402657	0.080249361	0.006101403
	215	4.7411	0.071958663	0.578738054	0.080249361	0.006089881
	235	4.6111	0.069985571	0.626939449	0.08072162	0.006069474
	260	4.4386	0.067382608	0.692875247	0.079956619	0.006090985
	275	4.3319	0.065747977	0.735720259	0.080134579	0.006090261
	285	4.2581	0.064627868	0.765745573	0.079663183	0.006091821
	315	4.0271	0.061121853	0.863375722	0.084235771	0.006076669
	325	3.9469	0.059904589	0.898610754	0.083425656	0.006079356
	343	3.798508	0.05765235	0.965696849	0.085161663	0.006067043
	360	3.6536	0.05545299	1.033648406	0.087134712	0.006069131
	375	3.5219	0.053454096	1.682543789	0.086506383	0.008082404
	415	3.1531	0.047856586	4.411906057	0.082506289	0.006084561
	422	3.085928	0.046837074	5.2226978	0.081855627	0.006110249
	445	2.52	0.038247628	9.091190054	0.074086484	0.006138798
	460	1.947	0.029550846	13.05024371	0.065413966	0.006097756
Low.piston.speed.setting	90	5.6945	0.086429016	0.245318161	0.077264092	0.006091492
	145	5.484675	0.08324437	0.304008598	0.079762526	0.006088997
	170	5.3713	0.081523606	0.335142557	0.080516202	0.006069405
	195	5.246675	0.079632094	0.3694699	0.086423894	0.006074261
	215	5.138875	0.077995945	0.399436972	0.084961429	0.006074261
	235	5.023875	0.076250518	0.431840361	0.085228964	0.006073373
	260	4.87	0.07391506	0.476065755	0.084769948	0.006074926
	275	4.772275	0.072431826	0.504746329	0.084999383	0.006074135
	285	4.704875	0.071408854	0.524820335	0.084499871	0.006076091
	315	4.491875	0.068176018	0.589960736	0.089338622	0.006059648
	325	4.417275	0.067043767	0.613423376	0.088487312	0.006062547
	343	4.278459	0.064936869	0.658031994	0.090140979	0.006057045
	360	4.142	0.062865745	0.703139977	0.092353564	0.006049675
	375	4.017275	0.060972715	0.882114227	0.092179115	0.006050257
	403	3.773619	0.057274594	1.443926701	0.089895229	0.006057863
	415	3.664875	0.055624118	1.783479583	0.089161011	0.006060306
	422	3.600244	0.054643173	2.017312697	0.088742101	0.006061699
	445	2.8695	0.04355221	3.023957118	0.0793254	0.006092939
	460	2.205	0.033466675	3.937579828	0.069549734	0.00612252
	475	1.5405	0.02338114	5.12723372	0.058344121	0.006161974
	490	0.876	0.013295604	6.676315595	0.045318582	0.006204445
	505	0.2115	0.003210069	8.693418784	0.030520403	0.006252347
Low.shot.size.setting	90	5.4444	0.082633091	0.265922274	0.074738464	0.006108098
	145	5.2002	0.078926713	0.344365662	0.076780753	0.006101353
	170	5.0812	0.077120575	0.387301734	0.077440345	0.006099173
	195	4.9572	0.075238549	0.435591446	0.082675434	0.006081843
	215	4.8544	0.073670289	0.478522898	0.081481552	0.00608558
	235	4.7484	0.07206946	0.526585994	0.081858631	0.006084551
	260	4.6114	0.069990125	0.59122938	0.081731097	0.006084973
	275	4.5268	0.068706097	0.634415476	0.082142553	0.00608361
	285	4.4694	0.067834901	0.664944823	0.081766605	0.006084855
	315	4.2924	0.065148461	0.765633859	0.087070712	0.006067256
	325	4.2318	0.064228696	0.802477698	0.086402198	0.006069477
	343	4.120704	0.062542522	0.873321767	0.088467006	0.006062614
	360	4.0134	0.060913902	0.945963575	0.090980846	0.006054248
	375	3.9168	0.059447743	1.682543789	0.091029965	0.006054085
	403	3.731664	0.056637817	3.303910638	0.088366825	0.006059621
	415	3.6504	0.055404422	4.411906057	0.088973122	0.006060931
	422	3.602464	0.054676867	5.2226978	0.08877177	0.0060616
	445	2.979	0.045214168	9.091190054	0.080963719	0.006087513
	460	2.289	0.034741596	13.05024371	0.070894284	0.006120773
	490	0.909	0.013796466	26.89140741	0.045931112	0.006202455
	505	0.219	0.003323901	38.602105	0.030672521	0.006251856
High.shot.size.setting	90	5.7002	0.086615529	0.249697	0.077324694	0.006099555
	145	5.495875	0.083414359	0.314583072	0.078888716	0.006091074
	170	5.385	0.081731539	0.349410756	0.080674103	0.006088474
	195	5.262875	0.079877971	0.388094234	0.08665048	0.006068652
	215	5.157075	0.078272178	0.422102502	0.085202806	0.006073459
	235	5.046075	0.076557106	0.459090878	0.085496443	0.006072485
	260	4.8927	0.074259592	0.509917109	0.085047437	0.006073975
	273	4.796475	0.072799125	0.543075407	0.085300754	0.006073134
	285	4.730075	0.07179133	0.566370343	0.084711375	0.006075091
	315	4.520075	0.068604028	0.642423781	0.089692861	0.006058536
	343	4.309459	0.065407376	0.722594481	0.09047421	0.006055935
	360	4.1747	0.063362053	0.776074236	0.092702449	0.006048513
	375	4.051475	0.06149179	0.826539691	0.092570101	0.006048954
	403	3.810619	0.057836167	0.929687196	0.090361100	0.006056312
	422	3.639144	0.055233582	1.006916713	0.089261901	0.00605997
Low.gas.pressure.setting	90	5.7078	0.086630879	0.207027526	0.077405707	0.006099288
	145	5.5405	0.084004831	0.271063133	0.080392919	0.006089405
	170	5.4438	0.082623984	0.306387842	0.081364729	0.006086187
	195	5.33505	0.080973417	0.346316036	0.087681572	0.006052225
	215	5.23905	0.079516364	0.3819737	0.086317859	0.006069757
	235	5.13505	0.077937891	0.421302778	0.086735312	0.00606837
	260	4.9938	0.075784605	0.476206583	0.08636352	0.006069605
	275	4.90305	0.074416681	0.512526162	0.086670001	0.006068587
	285	4.84005	0.07346049	0.538265406	0.086069222	0.006070583
	315	4.63905	0.070409786	0.623501866	0.091132239	0.006053744
	325	4.56805	0.069332174	0.654814349	0.090267616	0.006056623
	343	4.43521	0.067315978	0.715192515	0.091840359	0.006051386
	360	4.3038	0.065321486	0.7731973	0.094079197	0.006043922
	375	4.18305	0.063468787	0.915821573	0.094073524	0.006043941
	403	3.94561	0.05988501	1.503305391	0.092059776	0.006050654
	415	3.83905	0.058267682	1.859051252	0.091420321	0.006052785
	445	3.061	0.046458727	3.161562217	0.082189632	0.006083454
	460	2.365	0.035895096	4.122939079	0.072109944	0.006116768
	490	0.973	0.014767834	7.011602506	0.047118488	0.006198594

	505	0.277	0.004204204	9.143710609	0.031848498	0.006248063
	525			13.02754911	0.021879	0.00628015
High.gas.pressure.setting	90	5.4306	0.082423639	0.363448865	0.074606712	0.006108533
	145	5.13855	0.077991012	0.433390259	0.076191349	0.0061033
	170	4.9938	0.07579405	0.469486063	0.076614115	0.006101904
	195	4.84155	0.073483256	0.508588181	0.08133739	0.006086277
	215	4.71435	0.071552662	0.542201994	0.079971258	0.006090801
	235	4.58235	0.069549214	0.578037424	0.080077233	0.00609045
	260	4.4106	0.066942456	0.626180466	0.079675911	0.006091779
	275	4.30395	0.065323762	0.65697017	0.079865561	0.006091151
	285	4.23135	0.064221866	0.678333201	0.079415053	0.006092642
	315	4.00635	0.060806899	0.746601419	0.084024081	0.006073371
	325	3.92895	0.059632151	0.77096136	0.083247328	0.006079947
	343	3.786606	0.057471706	0.816672073	0.085042491	0.006073992
	360	3.6486	0.055377102	0.862330505	0.087081207	0.006067221
	375	3.52395	0.05348521	1.37867602	0.086529895	0.006069053
	415	3.17835	0.048239821	3.856650699	0.082835189	0.006081314
	422	3.115896	0.047291918	4.616787479	0.082257597	0.006083228
	443	2.587	0.03926453	8.337797663	0.075091949	0.006106931
	460	2.017	0.03061328	12.25947506	0.06653698	0.006135109
	475	1.447	0.021962031	18.02571073	0.056728409	0.006167258
	490	0.877	0.013310782	26.50409137	0.045337146	0.006204385
	505	0.307	0.004659533	30.97027252	0.032456482	0.006246101
Fourier and Capillary Number analysis for the polycarbonate data from the second set of experiments						
	Thermal diffusivity (m^2/s)					
	n					
	Tb(K)					
	T* (Pa)					
	B (Pa-sec)					
	B(m)					
	Rowlength (mm)	Representative shear rate (1/sec)	Temperature.(K)	Zero.shear.viscosity.(Pa-sec)	Viscosity (pa-sec)	
high.delay.time.setting	90	141.5760345	619.4935291	233.0315609	224.1389427	
	145	184.6474712	619.8142299	231.2733148	220.0794617	
	170	208.2969686	620.2889969	226.6980147	216.4660321	
	195	235.8150067	620.5233928	227.4386014	213.8907528	
	215	259.2813728	620.5408584	227.3450745	212.5972374	
	235	285.421266	620.6933715	226.5302341	210.5636111	
	260	321.6145699	620.8207368	225.8522961	208.1935138	
	275	345.6148342	620.805115	225.9353232	207.1021474	
	285	362.4055447	620.7913491	225.9872456	206.3420078	
	325	439.8872112	620.4039144	226.0795732	204.4175698	
	343	479.7896542	619.9685709	230.432517	204.4481664	
	360	521.1118366	619.7160242	231.8101223	203.6264375	
	375	637.3220337	619.7281665	231.7436738	198.4169866	
	403	1285.608508	620	230.2617279	173.3850291	
	415	1736.828522	620	230.2617279	160.1008102	
	422	2070.082041	620	230.2617279	151.5967861	
	445	3668.242869	620	230.2617279	121.1826174	
	460	5313.105199	620	230.2617279	100.7328855	
	475	7689.447006	620	230.2617279	81.20626009	
	490	11117.39849	620	230.2617279	63.65084791	
	505	16058.31531	620.7861928	226.0359383	48.46886224	
	525	26416.49333	621.9833333	219.7695418	32.52045923	
Base.setting	90	140.8093576	621.4028931	222.7828267	214.6693499	
	145	177.5870048	621.4826556	222.365978	212.3491981	
	170	197.3035821	621.8344108	220.5382113	209.6789305	
	195	219.7338224	621.6267096	221.6153795	209.5224074	
	215	238.8089318	621.9281912	220.0538045	207.1857718	
	235	259.7506227	622.0015459	219.6757463	205.8368377	
	260	288.4353486	622.1630961	218.8457446	203.7488514	
	275	307.2173405	622.1757577	218.7808436	202.8166103	
	285	320.3083965	622.2071051	218.6202569	202.0741626	
	315	364.2193753	621.9197305	220.0974568	201.3213552	
	325	379.6679742	622.12423	219.0451019	199.7451406	
	343	409.8614567	621.7951787	220.7412175	199.8046553	
	360	440.660853	621.523319	222.1538074	199.5905675	
	375	582.7364427	621.5314135	222.1116	193.4905194	
	403	1021.291061	621	224.9020212	178.9273527	
	415	1299.125416	621	224.9020212	169.8747979	
	422	1494.830773	621	224.9020212	164.0684408	
	445	2365.422958	621	224.9020212	142.6341707	
	460	3182.440685	621	224.9020212	127.245886	
	475	4278.877031	621	224.9020212	111.3074315	
	490	5748.441222	621	224.9020212	95.45732445	
Low.delay.time.setting	90	144.6575778	624.2654468	208.3633229	201.0512643	
	145	182.4760951	624.3090544	208.1520224	199.1051853	
	170	202.7749657	624.6793282	206.3676503	196.5633166	
	195	225.6509781	624.2360273	206.5060133	197.4794461	
	215	245.3346399	624.7559354	206.0006523	194.4794461	
	235	266.867384	624.8134107	205.7257958	193.2094627	
	260	296.4013634	624.9452275	205.0970003	191.4184998	
	275	315.696014	624.6940999	205.3406303	190.833743	
	285	329.1829152	624.9046207	205.2904703	190.2380812	
	315	374.0418555	624.2984556	208.2033565	190.8894211	
	343	420.840374	624.0921314	209.205539	189.8291179	
	375	481.3162218	623.8226093	210.5229714	188.4949755	
	403	540.2217262	624	209.654822	185.5216293	
	415	567.7407381	624	209.654822	184.4811776	
	422	584.3738388	624	209.654822	183.8588506	
	445	642.2682988	624	209.654822	181.729886	
High.piston.speed.setting	90	180.4174974	621.4755995	222.3819332	212.2171191	
	145	225.040311	621.5807584	221.8544965	209.4706973	
	170	246.7762782	621.9370615	220.0080489	206.6599866	
	215	296.4028133	622.0303642	219.5274238	203.872336	
	235	323.2774906	622.0997485	219.1707821	202.4088648	
	260	357.1961889	622.2553635	218.3732999	200.1878503	
	275	379.3206546	622.2618507	218.3401268	199.1720662	
	285	394.6999027	622.2892983	216.1998326	198.3760108	
	315	446.1325569	621.9825895	219.7733733	197.4216272	
	325	464.1343232	622.1843879	218.7366194	195.8112231	
	343	499.2574557	621.8414412	220.501855	195.7280294	
	360	534.9650566	621.5564155	221.9812892	195.3802112	
	375	870.5013328	621.5123864	222.0022827	182.3305443	
	415	2277.616662	622	219.6837059	142.308019	
	422	2695.207272	622	219.6837059	134.0038019	
	445	4671.882138	622	219.6837059	105.3224792	
	460	6675.209513	622	219.6837059	86.79964884	
Low.piston.speed.setting	90	126.283588	620.7254099	226.3594833	218.7978286	
	145	156.7082398	620.757894	226.1865061	216.9631041	
	170	172.8277441	621.0916873	224.417744	214.4743416	
	195	191.142176	620.9815463	224.999634	214.0386955	
	215	206.4830863	621.1543007	224.087712	212.4163021	
	235	223.2661869	621.2206178	223.7387588	211.2436204	
	260	246.0682574	621.3425125	223.0989738	209.5265369	
	275	260.9266261	621.3369482	223.1281339	208.8145619	
	285	271.2164493	621.3461855	223.0797279	208.2652955	
	315	305.7069706	621.0205224	224.7935223	208.0628233	
	325	317.7131256	621.1511088	224.1045229	206.8904787	
	343	341.1276607	620.7924553	226.0026332	207.3750666	
	360	364.955035	620.5063195	227.5300697	207.5092061	
	375	457.8051402	620.457154	227.6327966	203.2322684	
	Flowlength.(mm)	Representative Shear Rate (1/sec)	Temperature (K)	Zero shear viscocity (Pa-sec)	Viscosity (Pa-sec)	
High.delay.time.setting	403	748.437216	621	224.9020212	188.9150356	
	415	924.0665684	621	224.9020212	182.3499671	
	422	1044.981235	621	224.9020212	178.1145939	
	445	1556.398315	621	224.9020212	162.272862	
	450	2018.546285	621	224.9020212	150.4248872	
	475	2512.720342	621	224.9020212	137.5784329	
	490	3378.808251	621	224.9020212	124.0497396	
	505	4365.934212	620.6697842	226.6560332	110.6534082	
Low.shot.size.setting	90	136.7031069	621.3781395	222.9123724	215.0098757	
	145	177.2243275	621.4335181	222.6226714	212.6025365	
	170	199.3921862	621.7487668	220.9816501	209.9749693	
	195	224.8917139	621.5078415	222.2345378	209.8176682	
	215	245.8963706	621.742589	221.0136761	207.6412346	
	235	271.2861056	621.7606151	220.9202431	206.3736096	
	250	305.0893107	621.8371939	220.5238182	204.4182602	
	275	327.4477973	621.7970596	220.7314798	203.5488873	
	285		621.787173	220.7826692	202.8646967	
	315	396.2401541	621.3900968	222.8497842	202.1564379	
	325	415.1560503	621.5152773	222.1957485	200.7621919	
	343	452.3181461	621.104762	224.3487821	200.8443967	
	360	490.6183862	620.753442	226.2102041	200.6185069	
	375	872.6649144	620.6767818	226.6187079	185.378243	
	403	1712.034378	621	224.9020212	158.1016133	
	415	2285.686084	621	224.9020212	144.3488424	
	422	2705.416132	621	224.9020212	135.778906	
	445	4589.329523	621	224.9020212	106.3551199	
	450	6694.867386	621	224.9020212	87.49817225	
	490	13613.80696	621	224.9020212	54.72172358	
	505	19387.93916	620.62392	226.9008842	41.95363402	
High.shot.size.setting	90	128.5419245	620.711677	226.4326557	218.7404814	
	145	162.1702153	620.7344855	226.3111407	216.7813743	
	170	180.2011031	621.0551178	224.5579589	214.2128719	
	195	200.8050408	620.9579685	225.1244215	213.6455136	
	215	218.2284849	621.1231995	224.2515757	211.9594937	
	235	237.3897081	621.1883972	223.9082228	210.678798	
	260	263.6065623	621.3068208	223.2860923	208.8207347	
	275	280.7869353	621.2991521	223.3263194	208.0113341	
	285	292.7368496	621.3063966	223.2883173	207.3960859	
	315	332.9534793	620.9735323	225.0420402	206.9555244	
	343	374.664939	620.7348244	226.309336	206.0278066	
	360	402.8879684	620.4468308	227.8491013	205.9631602	
	375	429.0552287	620.4256759	227.9626751	204.8270288	

	403	482.0124549	620	230.2617279	204.2104426
	422	521.738301	620	230.2617279	202.3962639
Low.gas.pressure.setting	90	106.5807157	620.6934458	226.5298382	220.0655924
	145	139.7736234	620.6445152	226.7908995	218.4510244
	170	158.0723439	620.9556501	225.1366965	215.9220044
	195	179.2895539	620.8604917	225.6411619	215.2486408
	215	197.6022155	620.9928594	224.9397863	213.6460889
	235	217.9976996	621.0557756	224.6072823	212.2897991
	260	245.3568268	621.1630135	224.0418305	210.3454672
	275	265.1905846	621.1488547	224.1163955	209.4698922
	285	278.4169939	621.1495112	224.1129375	208.8122251
	315	323.402494	620.7896197	226.0177125	208.2473204
	325	339.4824148	620.8965831	225.4496786	206.9869094
	343	371.1058354	620.5136432	227.4908289	207.1807918
	360	403.8410824	620.2252306	229.0420028	206.895832
	375	475.7954847	620.2036848	229.1583645	203.6230072
	403	780.1435266	620	230.2617279	191.4376574
	415	964.4190406	620	230.2617279	184.3963445
	445	1631.853532	620	230.2617279	162.9940068
	460	2116.482004	620	230.2617279	150.4876274
	490	3551.842994	620	230.2617279	122.9631642
	505	4595.224498	620.2708922	228.7956214	108.3670307
	525		621.9833333	219.7695418	88.05589472
High.gas.pressure.setting	90	186.8254632	621.4170117	222.7089763	212.1835968
	145	222.9687795	621.6005	221.7517307	209.4824801
	170	241.5944825	621.9820912	219.7759405	206.8032505
	195	262.3881187	621.7736957	220.8524711	206.7443216
	215	279.5222191	622.1027526	219.1553556	204.4382759
	235	298.0136771	622.1766573	218.7762334	203.2405991
	260	322.7639662	622.3330017	217.976661	201.4103111
	275	338.6693784	622.3357239	217.9627685	200.6755461
	285	349.565489	622.3600438	217.8387003	200.0776586
	315	385.7884557	622.0337904	219.5097971	199.8591549
	325	398.164557	622.230996	218.497958	198.4707784
	343	422.1857688	621.8711956	220.348061	198.9361643
	360	446.2863448	621.5687321	221.9171263	199.1503925
	375	713.7799297	521.5473046	222.0287654	188.1991543
	415	1991.326792	622	219.6837059	148.6597665
	422	2383.06236	622	219.6837059	140.1103581
	445	4287.044529	622	219.6837059	109.8652124
	460	6274.501659	622	219.6837059	89.94760021
	475	9177.617275	622	219.6837059	71.32549266
	490	13413.55303	622	219.6837059	54.95727593
	505	19590.88754	622	219.6837059	41.32224819

Fourier and Capillary number analysis for the polycarbonate data...

Thermal Diffusivity (m^2/s)	
n	
Tb(K)	
T*(Pa)	
B (Pa-sec)	
R (m)	

	Rowlength.(mm)	Surface Tension (N/m)	Ca*Tau	Unmodified Fractional Coverage	Adjusted Fractional Coverage
High.delay.time.setting	90	1	61.38529	0.697294689	0.612305664
	145	1	78.48384	0.724263224	0.636329818
	170	1	87.03738	0.723513367	0.634638557
	195	1	96.96688	0.735492284	0.639207824
	215	1	106.0917	0.723607207	0.629374291
	235	1	115.6643	0.724449991	0.630122225
	260	1	128.9432	0.727618456	0.634400592
	275	1	137.8431	0.717437048	0.624271564
	285	1	144.0901	0.726801285	0.634653995
	325	1	172.9615	0.725024622	0629697292
	343	1	188.5991	0.720672799	0.624568812
	360	1	203.8106	0.728904796	0.630964354
	375	1	242.871	0.720264006	0.622224677
	403	1	428.5787	0.722674305	0.626599134
	415	1	534.8377	0.718447447	0.623043945
	422	1	603.7039	0.718957268	0.623865945
	445	1	859.5924	0.70168097	0.616004956
	460	1	1041.203	0.66145734	0.586886525
	475	1	1223.11	0.648457964	0.586610702
	490	1	1396.986	0.647927231	0.60090148
	505	1	1550.11	0.624885193	0.594750209
	525	1	1718.194	0.596755298	0.574876298
Base.setting	90	1	58.80275	0.670739472	0.596085116
	145	1	73.28249	0.703137029	0.626534552
	170	1	80.3721	0.691550379	0.614425569
	195	1	89.20351	0.698364593	0.616318619
	215	1	95.93832	0.701241289	0.620579037
	235	1	103.6659	0.702084104	0.621314768
	260	1	113.9748	0.702734839	0.622427186
	275	1	120.8309	0.695427813	0.614964781
	285	1	125.5526	0.700071162	0.620106977
	315	1	141.8815	0.69118994	0.60668815
	325	1	146.8089	0.697962535	0.614302011
	343	1	158.3852	0.696915367	0.611566536
	360	1	169.9251	0.695680165	0.608400996
	375	1	217.9243	0.697149418	0.610545835
	403	1	353.7884	0.685426476	0.601956019
	415	1	427.5113	0.697779619	0.615366148
	422	1	475.2794	0.691462878	0.609743872
	445	1	656.0272	0.652087133	0.576552653
	460	1	791.1918	0.645164384	0.576563253
	475	1	935.6253	0.62876281	0.572406124
	490	1	1084.743	0.619778545	0.575314121
Low.delay.time.setting	90	1	56.77421	0.65467158	0.588461557
	145	1	70.83516	0.665676326	0.595140182
	170	1	77.67293	0.665864314	0.594440691
	195	1	86.67424	0.663917007	0.588985106
	215	1	92.7868	0.665206569	0.590952632
	235	1	100.3132	0.66760097	0.593108795
	260	1	110.3861	0.664453562	0.590030529
	275	1	117.205	0.659576701	0.585032839
	285	1	121.851	0.653769429	0.579533241
	315	1	138.6866	0.648133579	0.570655252
	343	1	155.1274	0.655552792	0.577547121
	375	1	176.1954	0.651622842	0.573855303
	403	1	195.0574	0.648423281	0.574620285
	415	1	203.9897	0.648282304	0.575804331
	422	1	209.3674	0.633809094	0.562303589
	445	1	227.9313	0.628643675	0.561110918
High.piston.speed.setting	90	1	74.49296	0.696792714	0.622394394
	145	1	91.62265	0.704865477	0.628606809
	170	1	99.89999	0.707882141	0.631116543
	215	1	117.9887	0.708855373	0.628606012
	235	1	126.8981	0.705523156	0.625150994
	260	1	138.7052	0.699458032	0.619501413
	275	1	146.5349	0.701641672	0.621507092
	285	1	151.9056	0.704577476	0.624914293
	315	1	170.449	0.699915757	0.615679986
	325	1	175.9581	0.702106235	0.618660579
	343	1	189.0139	0.702440258	0.617278595
	360	1	201.9544	0.68981505	0.602680338
	375	1	306.7791	0.70152703	0.615020646
	415	1	627.8496	0.676931134	0.594424844
	422	1	699.8563	0.637488469	0.555612842
	445	1	957.5076	0.623183955	0.54909747
	460	1	1132.757	0.613461083	0.548047117
Low.piston.speed.setting	90	1	53.67508	0.68239709	0.605132999
	145	1	65.95865	0.69807312	0.618310595
	170	1	71.87948	0.707196133	0.626679931
	195	1	79.0798	0.706231596	0.619807702
	215	1	84.84692	0.707991626	0.623030197
	235	1	91.22352	0.70932566	0.624096676
	260	1	99.74841	0.708581543	0.623820595
	275	1	105.3984	0.703773498	0.618774115
	285	1	109.3019	0.708160712	0.623750842
	315	1	122.7489	0.690731214	0.601372592
	325	1	126.9115	0.703222289	0.614734976
	343	1	136.4598	0.703554796	0.613413817
	360	1	145.908	0.702598996	0.610245432
	375	1	179.2741	0.702197434	0.610018318

	Rowlength.(mm)	Surface Tension (N/m)	Ca*Tau	Unmodified Fractional Coverage	Fractional Coverage
High.delay.time.setting	403	1	272.7795	0.692279374	0.602384145
	415	1	325.2174	0.697874714	0.608713703
	422	1	359.3128	0.697530677	0.608080576
	445	1	490.7062	0.660354829	0.589029429
	460	1	592.31	0.656144062	0.586594328
	475	1	705.3968	0.640846623	0.582502502
	490	1	828.1952	0.634127931	0.588809349
	505	1	961.9564	0.633314738	0.602794334
Low.shot.size.setting	90	1	57.17592	0.670580778	0.595842314
	145	1	73.21301	0.701208446	0.624427694
	170	1	81.32367	0.702841462	0.625401118
	195	1	91.39472	0.701430498	0.618755064
	215	1	99.36109	0.707964274	0.626482722
	235	1	108.4877	0.704267557	0.622408924
	260	1	120.8581	0.704437699	0.622706602
	275	1	129.1346	0.697208985	0.615066433
	285	1	134.8938	0.700352376	0.618565771

	315	1	154.7778	0.689056354	0.601985642
	325	1	161.1072	0.698702575	0.612300377
	343	1	175.4018	0.701758078	0.613291072
	360	1	189.7778	0.698799327	0.607818482
	375	1	311.907	0.699270899	0.608240933
	403	1	522.3536	0.686868841	0.597502016
	415	1	636.8535	0.695377846	0.606404724
	422	1	709.127	0.692602561	0.603830791
	445	1	966.8956	0.628149629	0.54718591
	460	1	1141.872	0.622848174	0.551953891
	490	1	1471.544	0.61980335	0.573872239
	505	1	1619.499	0.616109316	0.585436794
High.shot.size.setting	90	1	54.61884	0.672382415	0.595057721
	145	1	66.19575	0.699464312	0.619575596
	170	1	74.84828	0.701181268	0.620507165
	195	1	82.91459	0.701738315	0.615087834
	215	1	89.46863	0.703221545	0.618018739
	235	1	96.72071	0.705989659	0.620493216
	260	1	106.4813	0.701207358	0.616159921
	275	1	112.9658	0.700674887	0.615374134
	285	1	117.463	0.703816372	0.619104997
	315	1	132.9532	0.698529973	0.608637112
	343	1	148.8746	0.706125555	0.615651345
	360	1	159.8427	0.704326992	0.611624542
	375	1	169.2977	0.708604945	0.616034845
	403	1	189.6518	0.682532306	0.592171205
	422	1	203.7962	0.673952105	0.584690204
Low.gas.pressure.setting	90	1	45.55964	0.70186614	0.624460433
	145	1	59.21402	0.704966546	0.624573626
	170	1	66.15588	0.715469093	0.634104364
	195	1	74.54406	0.709629477	0.621947905
	215	1	81.60719	0.711614645	0.625296785
	235	1	89.43828	0.705424738	0.618689426
	260	1	100.1679	0.704353297	0.617989777
	275	1	107.3588	0.710230389	0.623560387
	285	1	112.3964	0.708774738	0.622705517
	315	1	129.8426	0.703280722	0.612148342
	325	1	135.538	0.711482142	0.621214526
	343	1	148.1742	0.712119017	0.620278658
	360	1	160.8242	0.714311394	0.620232198
	375	1	186.4823	0.710045123	0.615971599
	403	1	287.7893	0.705971812	0.613912036
	415	1	342.8023	0.703328336	0.611908215
	445	1	515.3157	0.699779951	0.617590319
	460	1	620.4521	0.707333813	0.635223869
	490	1	862.1688	0.662368545	0.615250058
	505	1	990.8768	0.641571248	0.60972275
	525	1	1147.152	0.637470269	0.615591269
High.gas.pressure.setting	90	1	77.11789	0.682723977	0.608117265
	145	1	90.78767	0.704219021	0.628027671
	170	1	97.09124	0.699624337	0.623210223
	195	1	105.1477	0.701748647	0.620411257
	215	1	110.8468	0.703458037	0.623487679
	235	1	117.4807	0.70243286	0.622355627
	260	1	126.1192	0.704290352	0.624614442
	275	1	131.8378	0.697302691	0.617437129
	285	1	135.7193	0.698056443	0.618641389
	315	1	149.2311	0.696005632	0.611981551
	325	1	153.0134	0.694936722	0.611689394
	343	1	162.4658	0.698370737	0.613328246
	360	1	171.7335	0.698107561	0.611026355
	375	1	259.641	0.69501785	0.608487955
	415	1	573.3288	0.682227046	0.599391857
	422	1	646.8597	0.681293654	0.599036357
	445	1	916.0339	0.641569264	0.566497315
	460	1	1102.71	0.613927755	0.547390775
	475	1	1285.693	0.610523635	0.553795225
	490	1	1456.593	0.61087492	0.565327781
	505	1	1610.339	0.60715379	0.574697308

18 Appendix K: Data for PS

Fourier.and.Capillary.number.analysis.using.polystyrene.data.from.the.second.set.of.experiments							
Thermal.Diffusivity.(m^2/s)	0.000000075						
n	0.2968						
Tb.(K)	11376						
T.(Pa)	17380						
B.(Pa-sec)	1.68E-06						
R.(m)	0.00635						
		7.5931			CMOLD.character		
High.piston.speed.setting	Rowlength.(mm)	Residual.Time.(sec)	Fourier.#	Ub.(m/sec)	Tmelt.(deg.K)	Solid.Layer.Fractional.Coverage	Adjusted.Radius.(m)
	90	4.9947	0.009290161	0.145890893	531.386667	0.04196873	0.006215321
	145	5.0332	0.009361771	0.283825336	531.2447108	0.04548924	0.006203891
	170	5.0107	0.009319921	0.384083138	531.3160248	0.04605992	0.006202036
	195	4.9632	0.00923157	0.519755773	531.4157354	0.04619392	0.0062016
	215	4.9072	0.00912741	0.662061886	531.4785738	0.04628032	0.006201319
	260	4.7227	0.00878424	1.141226722	531.5328812	0.04691939	0.006199241
	275	4.6432	0.008636369	1.368347429	531.4588205	0.04726624	0.006198113
	285	4.5852	0.008528489	1.544350998	531.3914606	0.04753564	0.006197237
	315	3.816	0.007097774	2.220214373	531.6054448	0.0444328	0.006207323
	325	3.542	0.006588133	2.505789253	531.6943954	0.0429436	0.006212158
	343	3.0488	0.005670779	3.111543938	531.6816369	0.04048792	0.006220123
	360	2.583	0.00480439	3.827086297	531.538905	0.0383563	0.006227028
	375	2.172	0.004039928	4.588732104	531.14346	0.0358664	0.006235084
	460		0	12.83407516	529.7277778		0.00635
	475		0	15.38824268	529.7277778		0.00635
	490		0	18.45072667	529.7277778		0.00635
Low.piston.speed.setting	90	5.7238	0.010646289	0.090423223	531.0197838	0.04627042	0.006210352
	145	6.0428	0.01123963	0.145111314	530.734156	0.05124396	0.006185161
	170	6.1078	0.011360531	0.179917889	530.7546387	0.05220368	0.006182032
	195	6.1228	0.011388431	0.223073459	530.8096124	0.05268768	0.006180453
	215	6.0988	0.011343791	0.264939402	530.842021	0.05295328	0.0061795 87
	235	6.0428	0.01123963	0.314662655	530.8851129	0.05364396	0.006177333
	260	5.9278	0.01102573	0.390138236	530.8880342	0.05378846	0.006176861
	275	5.8348	0.01085275	0.443856419	530.8348987	0.05405836	0.00617598
	285	5.7628	0.010718829	0.483717534	530.7882938	0.05424796	0.006175361
	315	5.4988	0.010227788	0.626094381	530.7435146	0.05419304	0.006175541
	325	5.3948	0.010034348	0.682321619	530.7655868	0.05368984	0.006177183
	343	5.18744	0.009548658	0.796558943	530.6446103	0.053105896	0.00617098 9
	360	4.9678	0.009240126	0.921959346	530.429973	0.05290358	0.006178749
	375	4.7548	0.008843946	1.048904042	529.994114	0.05187976	0.006183088
	415	2.866589073	0.005331866	1.479561633	529.7277778	0.005485263	0.00633256
	445	1.052452176	0.001957565	1.915054056	529.7277778	0.021424007	0.006281611
	460	0.637706487	0.001186136	2.178738087	529.7277778	0.017718015	0.006293494
	475	0.386401941	0.000718709	2.478728804	529.7277778	0.015507806	0.00630057
	490	0.234130377	0.000435483	2.820025281	529.7277778	0.01510691	0.006301853
Base.Setting	90	5.2493	0.009763718	0.101990478	531.2585522	0.04347087	0.006210447
	145	5.41155	0.010065503	0.180707455	531.0533792	0.047645835	0.006196878
	170	5.4253	0.010091078	0.234364936	531.103874	0.04838168	0.006194484
	195	5.40155	0.010046903	0.303954936	531.1866098	0.04864868	0.006193615
	215	5.35555	0.009961343	0.37423332	531.2390652	0.04879106	0.006193151
	235	5.28555	0.009831143	0.460760993	531.282972	0.049327635	0.006191404
	260	5.1643	0.009605617	0.597574794	531.2965831	0.04943651	0.00619105
	275	5.07355	0.009436822	0.698461077	531.2334892	0.049719235	0.006190129
	285	5.00555	0.009310342	0.775012729	531.1761573	0.049931635	0.006189437
	325	4.67355	0.00869282	1.174830845	531.1271494	0.04950659	0.006190821
	343	4.49283	0.008356928	1.416691431	530.9614267	0.049007697	0.006192446
	360	4.3043	0.008006014	1.690668774	530.7385005	0.04885623	0.006192939
	375	4.12355	0.007669818	1.976097964	530.2750203	0.04796601	0.006195837
	445	0.674436903	0.001254455	4.092369635	529.7277778	0.021424007	0.006281611
	460	0.402573676	0.000748789	4.783268862	529.7277778	0.017718015	0.006293494
	475	0.240297593	0.000446954	5.590809982	529.7277778	0.015507806	0.00630057
	490	0.143434448	0.000266789	6.534685204	529.7277778	0.01510691	0.006301853
High.delay.time.setting	90	7.3771	0.013721433	0.098939402	530.1878433	0.05602489	0.006169557
	145	7.56685	0.014074369	0.19354452	529.963444	0.059931045	0.006156779
	170	7.5931	0.014123194	0.262567334	529.9946107	0.06052136	0.006154846
	195	7.58185	0.014102269	0.356205411	530.046967	0.06085836	0.006153742
	215	7.54585	0.014035309	0.454640757	530.0690069	0.06105676	0.006153092
	235	7.48585	0.013923709	0.580278152	530.1269344	0.061869345	0.006150429
	275	7.29385	0.013566588	0.945304938	530.0709401	0.062374945	0.006148771
	285	7.23085	0.013449408	1.067962131	530.0363586	0.062615845	0.006147981
	315	7.00585	0.013030907	1.539953594	529.9716036	0.062993393	0.006146938
	325	6.91885	0.012869087	1.739768B9	530.0015805	0.06252933	0.006148265
	343	6.74713	0.012549687	2.167018549	529.8883167	0.062308067	0.00614899
	360	6.5671	0.012214483	2.66646142	529.6862985	0.06265931	0.006147839
	375	6.39385	0.011892585	3.201925292	529.2647368	0.06204187	0.006149863
	445	0.483969126	0.000900184	7.521399937	529.7277778	0.021424007	0.006281611
	460	0.234867404	0.000436854	9.031805413	529.7277778	0.017718015	0.006293494
	475	0.113979786	0.000212003	10.84552207	529.7277778	0.015507806	0.00630057
	490	0.055313728	0.000102884	13.02345917	529.7277778	0.01510691	0.006301853
	525	0.010237128	1.90411E-05	19.96043935	529.7277778	0.01171449	0.006312697
Low.delay.time.setting	90	3.8644	0.007187798	0.15143272	531.9554339	0.03529996	0.006236916
	145	3.8864	0.007228718	0.227497996	531.8246475	0.03895248	0.006225097
	170	3.8564	0.007172918	0.273729784	531.9066801	0.03959584	0.006223013
	195	3.8014	0.007070618	0.329356724	532.0230082	0.03968784	0.006222715
	215	3.7394	0.006955298	0.381893369	532.1024125	0.03974064	0.006222544
	235	3.6614	0.006810218	0.442810287	532.1363004	0.04006998	0.006221477
	275	3.4574	0.006430777	0.595345215	532.0797054	0.04050718	0.00622006
	285	3.3964	0.006317317	0.64107178	532.0003639	0.04075948	0.006219242
	325	3.1124	0.005789076	0.861901876	531.9097539	0.04045192	0.006220239
	343	2.96372	0.00551253	0.984704338	531.7228922	0.039985948	0.006221749
	360	2.8114	0.005229214	1.116709207	531.432699	0.03974954	0.006222516
	375	2.6674	0.004961274	1.24780518	530.9233007	0.03893788	0.006225145
	445	0.309	0.000574741	2.094649324	529.7277778	0.021424007	0.006281611
	460		0	2.340550486	529.7277778	0.017718015	0.006293494
	475		0	2.615319192	529.7277778	0.015507806	0.00630057
High.gas.pressure.setting	90	5.2183	0.009706057	0.098665765	531.2741514	0.04328797	0.00621104
	145	5.3833	0.010012958	0.190861005	531.0676652	0.04748481	0.006197402
	170	5.4183	0.010078058	0.257635408	531.1074559	0.04834248	0.006194611
	195	5.4283	0.010096658	0.347771425	531.1726276	0.04878948	0.006193127
High.piston.speed.setting	215	5.4183	0.010078058	0.442104126	531.2055441	0.04914248	0.006192007
	235	5.3923	0.010029698	0.562024498	531.2268856	0.04993611	0.006189423
	275	5.2923	0.009843698	0.908273404	531.1189517	0.05096611	0.006186067
	285	5.2573	0.009778598	1.024075403	531.0472109	0.05136661	0.006184781
	315	5.1283	0.009538657	1.467837398	530.9332847	0.05204414	0.006182552
	343	4.97542	0.0092543	2.054002153	530.7474188	0.051854978	0.006183169
	360	4.8673	0.00905196	2.518819048	530.4767055	0.05229053	0.006181749
	375	4.7623	0.008857896	3.015573899	529.9907765	0.05192626	0.006182936
	445	0.880841874	0.001638369	6.985175795	529.7277778	-0.008026316	0.006375433
	460	0.530530859	0.000986789	8.362773746	529.7277778		0.00635
	475	0.319538614	0.000594343	10.01205793	529.7277778		0.00635
	490	0.192458034	0.000357973	11.9866096	529.7277778		0.00635
Low.gas.pressure.setting	90	5.5839	0.010386076	0.131866836	531.0901815	0.04544501	0.006204034
	145	5.76265	0.010718555	0.219928857	530.8738279	0.049647105	0.006190364
	170	5.7839	0.010758076	0.277495333	530.9203764	0.05038984	0.006187944
	195	5.76765	0.01072785	0.350129861	530.9203764	0.05069884	0.006186938
	215	5.72765	0.01065345	0.421704198	530.9952493	0.05087484	0.006186364
	235	5.66365	0.01053441	0.507909924	531.0402894	0.051482805	0.006184382
	260	5.5499	0.010322835	0.640855573	531.0843183	0.05163443	0.006183888
	275	5.46365	0.010162409	0.736790887	531.0902485	0.051942805	0.006182883
	325	5.07865	0.009446308	1.172861547	530.9240728	0.05185617	0.006183165
	343	4.90333	0.009120212	1.386730527	530.7823753	0.051429647	0.006184558
	360	4.7199	0.008779032	1.624254332	530.5452465	0.05139139	0.00618468
	375	4.54365	0.008451206	1.867403265	530.0880758	0.05057063	0.006187355
	445	0.439628467	0.000800971	3.58066075	529.7277778	0.021424007	0.006281611
	460	0.193300446	0.00035954	4.116687511	529.7277778	0.017718015	0.006293494
	475	0.086768677	0.000161319	4.732950714	529.7277778	0.015507806	0.00630057
	490	0.038948712	7.24447E-05	5.441467784	529.7277778	0.01510691	0.006301853
High.shot.size.setting	90	5.4207	0.010082522	0.095081641	531.1723038	0.04448213	0.006207163
	145	5.58845	0.010394538	0.184977816	530.9639208	0.048854165	0.006193597
	170	5.5047	0.010424763	0.25031895	531.012075	0.04938632	0.006191213
	195	5.58345	0.010385238	0.338741034	531.0913307	0.04966732	0.006190298
	215	5.53945	0.010303398	0.431486361	531.1408258	0.04962092	0.006189798
	235	5.47145	0.010176917	0.549624822	531.1853002	0.050387265	0.006187953
	275	5.26345	0.009790037	0.891794659	531.1340576	0.050801665	0.006186602
	285	5.19645	0.009665416	1.006501668	531.0783783	0.051019765	0.006185892
	325	4.86845	0.009055335	1.633100939	531.029446	0.05063701	0.006187139
	343	4.68953	0.008722543	2.030497067	530.8860469	0.050168227	0.006188666
	360	4.5027	0.008375039	2.494231395	530.6462445	0.05006647	0.006188998
	375	4.32345	0.008041633	2.990619701	530.1860648	0.04920539	0.006191802
	445	1.5285	0.002343016	6.976034442	529.7277778	0.021424007	0.006281611
	475		0	10.02899727	529.7277778		0.00635
Low.shot.size.setting	90	4.9817	0.009265981	0.111046655	531.3932086	0.04189203	0.00621557
	145	5.07795	0.009445006	0.182171841	531.2220807	0.045744315	0.006203062
	170	5.0617	0.009414781	0.225137935	531.2899281	0.04634552	0.006201107
	195	5.00795	0.009314806	0.285702318	531.3923445	0.046444452	0.006200786
	215	4.93795	0.009184605	0.342047776	531.417703	0.048146315	0.00619525
	235	4.84395	0.009009765	0.409505537	531.5149887	0.046810515	0.006199595
	275	4.58395	0.008526164	0.586956331	531.4898438	0.046928515	0.006199212
	285	4.50395	0.008377364	0.642232523	531.4330768	0.047027515	0.006198743
	325	4.12395	0.007670562	0.920530757	531.4026639	0.04631891	0.006201194
	343	3.92163	0.007294246	1.08241583	531.2584016	0.045637617	0.006203409
	360	3.7127	0.006905696	1.261365854	531.0135945	0.04524747	0.006204676
	375	3.51395	0.00653596	1.443679619	530.5462923	0.04418649	0.006208123
	445	0.274495817	0.000510563	2.710668125	529.7277778	0.021424007	0.006281611
	475	0.072670536	0.000135167	3.550878882	529.7277778	0.015507806	0.00630057
	490	0.037391224	6.95478E-05	4.064114497	529.7277778	0.01510691	0.006301853
High.melt.temperature	90	5.0643	0.009419617	0.122735776	547.3741126	0.041140863	0.006218006
	145	5.0588	0.009409387	0.201347825	547.2759426	0.043838009	0.006209255
	170	5.0163	0.009330337	0.252152454	547.413458	0.043996412	0.006208744
	195	4.9488	0.009204786	0.315776247	547.4966586	0.044017922	0.006208671
	215	4.8768	0.009070866	0.378052805	547.5419642	0.044210332	0.006208045
	235	4.7888	0.008907186	0.452611383	547.5824666	0.044119605	0.006208344
	275	4.5648	0.008490545	0.648741208	547.547024	0.044457465	0.006207243
	285	4.4988	0.008367785	0.709835947	547.4964025	0.044487181	0.006207146

	325	4.1948	0.007802344	1.017428742	547.453482	0.043838556	0.006209253
	343	4.03712	0.007509058	1.196354006	547.299055	0.042944074	0.006212156
	360	3.6763	0.007209932	1.394141207	546.9938623	0.042789991	0.006212656
	375	3.7248	0.006928142	1.595645894	546.504103	0.042238071	0.006214447
	445	0.823775185	0.001532225	2.996001612	548	0.021424007	0.006281611
	460	0.526840401	0.000979925	3.429034051	548	0.017718015	0.006293494
	475	0.336937569	0.000626705	3.924655606	548	0.015507806	0.00630057
	490	0.21548637	0.000400805	4.491912707	548	0.01510691	0.006301853
	505	0.137812995	0.000256333	5.141159326	548	0.01171449	0.006312697
Low.melt.temperature	90	5.3622	0.009973712	0.091223369	519.2441407	0.046313926	0.00620121
	145	5.6757	0.010556823	0.185658083	519.1282044	0.050929255	0.006186187
	170	5.7382	0.010673073	0.256314667	518.9377919	0.05378125	0.006176885
	195	5.7507	0.010696323	0.353861898	519.0029204	0.054248107	0.006175361
	215	5.7247	0.010647963	0.458017191	519.0789901	0.054475269	0.006174619
	235	5.6667	0.010540083	0.59282943	519.1320831	0.05430879	0.006175163
	260	5.5492	0.010321533	0.818445425	519.1582318	0.054355468	0.00617501
	275	5.4547	0.010145762	0.993174837	519.135743	0.054545279	0.00617439
	285	5.3817	0.010009982	1.129925203	519.0804635	0.054868847	0.006173334
	315	5.1147	0.009513361	1.662878693	519.0804635	0.055757346	0.006170431
	325	5.0097	0.009318061	1.892978357	519.0299691	0.055278226	0.006171997
	343	4.80054	0.008928142	2.387749654	518.971995	0.054869617	0.006173331
	360	4.5792	0.008517329	2.973237537	518.7990256	0.054296653	0.006175202
	375	4.3647	0.008118358	3.607992195	518.34412	0.05349073	0.006177833
	460	0.20394981	0.000379347	10.80113735	520	0.021424007	0.006281611
	475	0.10077335	0.000187439	13.10706554	520	0.017718015	0.006293494
	490	0.049792976	9.26151E-05	15.90828492	520	0.015507806	0.00630057
	525	0.00961067	1.78759E-05	24.98189691	520	0.01510691	0.006301853

High.piston.speed.setting	Rowlength.(mm)	Representative.shear.rate.(1/sec)	Zero-shear.Viscosity.(Pa-sec)	Viscosity.(Pa-sec)	Surface.Tension.(N/m)	Ca.#	
	90	73.70454396	3328.944336	447.7785881	0.01926192	3391.500801	
	145	143.6536513	3348.042341	295.5004855	0.019276244	4354.65613	
	170	194.4556723	3338.43321	242.7222173	0.019269048	4839.88665	
	195	263.1632261	3325.048345	190.70508	0.019258987	5361.776549	
	215	335.2309728	3316.643243	169.0459603	0.019252647	5810.370176	
	260	578.0468467	3309.397662	117.0608912	0.019247167	6936.602216	
	275	693.2127247	3319.282884	103.5585919	0.01925464	7356.698129	
	285	782.4877887	3328.301514	95.42703708	0.019261437	7650.994169	
	315	1123.104625	3299.744014	74.29212167	0.019239846	8563.239478	
	325	1266.577297	3287.951944	68.3182708	0.019230871	8887.545273	
	343	1572.767694	3289.640491	58.85159889	0.019232158	9519.027139	
	360	1929.821243	3308.595347	51.17847152	0.019246559	10168.47874	
	375	2310.89396	3361.737248	45.39134724	0.01928646	10813.49777	
	460	6346.298584	3559.772763	22.85260407	0.019429302	15226.52122	
	475	7609.304257	3559.772763	20.12984999	0.019429302	16061.62696	
	490	9123.666418	3559.772763	17.72986998	0.019429302	16983.19689	
Low.piston.speed.setting	90	45.76500653	3378.548515	597.7831693	0.019298939	2806.235314	
	145	73.66817648	3417.72617	452.54244495	0.019327759	3409.267006	
	170	91.38459706	3414.900225	396.1606459	0.019325692	3700.380102	
	195	113.3332208	3407.32827	345.8706702	0.019320145	4005.549056	
	215	134.6222276	3402.872983	309.8793076	0.019316875	4262.261664	
	235	159.9461705	3396.958907	277.2197577	0.019312527	4528.660882	
	260	198.3263043	3396.558379	241.0606627	0.019312232	4882.534053	
	275	225.6660679	3403.851559	221.6613287	0.019317594	5107.787902	
	285	245.956956	3410.262465	209.5742226	0.019322796	5262.960523	
	315	318.3423929	3416.434671	176.7123003	19326814	5743.901783	
	325	346.8393017	3413.390803	166.8291827	0.019324587	5909.647441	
	343	404.7830104	3430.110611	150.6527531	0.019336794	6230.105611	
	360	468.4579313	3459.995818	136.914808	0.019358451	6553.338647	
	375	532.672128	3521.562767	126.2332935	0.019402429	6874.008911	
	415	733.640625	3559.772763	101.8703232	0.019429302	7824.942653	
	445	957.281534	3559.772763	84.90132972	0.019429302	8441.039795	
	460	1087.033325	3559.772763	77.80046433	0.019429302	8800.100515	
	475	1235.31808	3559.772763	71.24368134	0.019429302	9168.024905	
	490	1405.123119	3559.772763	65.18770912	0.019429302	9543.75186	
Base.setting	90	51.56635676	3346.174924	555.5314651	0.019274847	2941.499013	
	145	91.56568529	3373.972837	393.8713757	0.019295549	3695.140093	
	170	118.8001971	3367.10819	334.2535271	0019290454	4006.952054	
	195	154.0971693	3355.893444	282.7367126	0.019282106	4461.612242	
	215	189.7406651	3348.804338	246.9028515	0.019276813	4796.991808	
	235	233.6771209	3342.883115	215.2976349	0.019272383	5150.096688	
	260	302.0802441	3341.049872	181.2337156	0.01927101	5622.528729	
	275	354.3008245	3349.557127	163.445994	0.019277376	5926.754125	
	285	393.1762985	3357.308016	152.5483963	0.019283161	6137.858899	
	325	595.877117	3363.9491	115.2758481	0.019288106	7030.951218	
	343	718.3609051	3383.781044	101.6944363	0.019302809	7479.505381	
	360	857.2181751	3417.126562	90.40635502	0.01932732	7935.200664	
	375	1001.470513	3481.74769	81.7414058	0.019374085	8385.925341	
	445	2045.66021	3559.772763	50.27014469	0.019429302	10680.35234	
	460	2386.506524	3559.772763	45.1727041	0.019429302	11217.63487	
	475	2786.278452	3559.772763	40.5645422	0.019429302	11773.93759	
	490	3256.012389	3559.772763	36.39827744	0.019429302	12348.26441	
High.delay.time.setting	90	50.35527037	3494.051063	573.4859359	0.019302882	2945.726835	
	145	98.709044	3525.939925	302.108523	0.019405524	3839.441226	
	170	133.9532216	3521.491924	314.7299333	0.019402379	4290.216052	
	195	181.7568915	3514.03364	258.2133622	0.019397096	4775.068911	
	215	232.0088861	3510.899163	219.9685962	0.019394872	5191.937639	
	235	296.2514458	3502.675389	186.9346924	0.019388027	5631.531862	
	275	482.7399571	3510.62437	134.7974665	0.019394077	6615.369017	
	285	545.4475208	3515.543434	124.1521669	0.019398166	6883.519953	
	315	786.6443906	3524.774815	96.82408047	0.019404	7740.899578	
	325	888.5229275	3520.498065	89.04414618	0.019401676	8042.616359	
	343	1106.594384	3536.687067	76.69590038	0.019413104	8628.497895	
	360	1361.891446	3565.76428	66.64371169	0.019433487	9225.605926	
	375	1634.840513	3627.286446	59.05305881	0.019476023	9816.439876	
	445	3759.735788	3559.772763	32.9289926	0.019429302	12858.12212	
	460	4506.220153	3559.772763	29.02343804	0.019429302	13608.92583	
	475	5405.056609	3559.772763	25.564065	0.019429302	14393.97666	
	490	6489.148761	3559.772763	22.49992151	0.019429302	15212.75161	
	525	9928.526638	3559.772763	16.71137318	0.019429302	17317.39856	
Low.delay.time.setting	90	76.23940468	3253.611617	434.7439082	0.019204532	3417.855089	
	145	114.7522125	3270.767642	338.0267346	0.019217728	3992.354014	
	170	138.1182165	3259.99541	300.1594239	0.019209451	4265.5443	
	195	166.1943453	3244.786073	266.088455	0.019197713	4549.807105	
	215	192.7097846	3234.44891	241.473899	0.019189702	4787.543473	
	235	223.4878086	3230.048218	219.0966319	0.019186282	5036.799729	
	275	300.5413375	3237.401336	180.3332706	0.019191993	5573.719969	
	285	323.6672994	3247.740613	171.8420782	0.019199996	5719.217249	
	325	435.091271	3259.592529	141.1356036	0.019208141	6315.312265	
	343	496.9617676	3284.183915	129.3519691	0.019227995	6612.708533	
	360	563.512764	3322.77704	119.2547227	0.019257276	6913.788692	
	375	629.4003458	3391.767471	111.3601256	0.019308704	7214.012905	
	445	1047.056145	3559.772763	79.8310601	0.019429302	8681.278408	
	460	1167.766054	3559.772763	74.05768118	0.019429302	8998.881675	
	475	1303.390302	3559.772763	68.65729528	0.019429302	9322.058197	
High.gas.pressure.setting	90	49.87100874	3144.07173	566.360203	0.019273273	2900.52087	
	145	96.7023823	3372.029136	380.6313632	0.019294108	3771.570188	
	170	130.5933706	3366.62182	314.7268455	0.019290093	4209.589596	
	195	176.3248608	3357.785848	259.2018628	0.019263517	4679.853364	
High.piston.speed.setting	215	224.1933727	3353.332666	221.4784296	0.019280196	5083.425044	
	235	285.1245485	3350.448896	186.9156264	0.019278042	5512.18198	
	275	461.0326251	3365.061371	137.1533979	0.019288933	6467.308636	
	285	519.9225506	3374.81244	126.5720888	0.019296171	6729.306363	
	315	745.4865521	3390.360932	99.21592795	0.019307667	7560.660996	
	343	1043.084358	3415.896044	79.01348726	0.01932642	8435.633083	
	360	1279.426322	3453.46482	68.88950978	0.019353735	9008.458487	
	375	1531.457114	3522.038799	61.22237918	0.019402766	9584.745744	
	445	3440.308021	3559.772763	35.02959305	0.019429302	12703.19167	
	460	4135.292844	3559.772763	30.81499541	0.019429302	13378.66789	
	475	4950.844395	3559.772733	27.17928644	0.019429302	14127.38837	
	490	5927.236872	3559.772763	23.96962319	0.019429302	14916.19272	
Low.gas.pressure.setting	90	66.74151124	3368.968137	477.7802565	0.019291836	3270.914614	
	145	111.5567077	3398.232261	348.9942023	0.019313464	3984.747878	
	170	140.8117613	3392.127309	300.7490353	0.019308969	4332.717222	
	195	177.6982162	3381.894396	258.5477221	0.019301414	4699.701614	
	215	214.0435299	3375.754859	228.8383082	0.01929687	5009.992453	
	235	257.8813987	3369.764925	202.3001457	0.019292427	5334.37217	
	260	325.40796	3768.959035	173.3342076	0.019291829	5766.932435	
	275	374.1820063	3377.260874	158.0081693	0.019297985	6043.996519	
	325	595.6758469	3391.62159	115.6057639	0.019308596	7039.974498	
	343	704.0657618	3411.077558	103.3618298	0.019322893	7441.366306	
	360	824.6438517	3443.910388	93.06449994	0.019346821	7847.629664	
	375	947.6821608	3508.189697	85.1000173	0.019392948	8250.270362	
	445	1789.874026	3559.772763	55.14423163	0.019429302	10250.95506	
	460	2053.930457	3559.772763	50.12974679	0.019429302	10713.80731	
	475	2358.749203	3559.772763	45.54109051	0.019429302	11190.14789	
	490	2711.299162	3559.772763	41.34105972	0.019429302	11678.79635	
high.shot.size.setting	90	48.09863301	3357.82969	579.4258443	0.01928355	2860.190434	
	145	93.77916446	3386.172075	388.5946617	0.019304575	3731.787412	
	170	126.95436	3379.599413	320.8913391	0.019299717	4170.154463	
	195	171.8248206	3368.78482	263.8873962	0.0192917	4640.736123	
	215	218.8871485	3362.094358	225.1901001	0.019286726	5044.484423	
	235	278.9003089	3356.070647	191.8099356	0.019282238	5473.156361	
	275	452.628927	3363.012098	138.8379125	0.019287409	6427.962883	
	285	510.9069827	3370.572354	128.027085	0.019293027	6389.85608	
	325	828.8058401	3377.231833	92.17349778	0.019297964	7814.829641	
	343	1030.23178	3396.830843	79.54964654	0.019312433	8385.73317	
	360	1265.453105	3429.884164	69.26508175	0.019336629	8969.154621	
	375	1516.609448	3494.302533	61.48541884	0.019383061	9546.270674	
	445	3487.122952	3559.772763	34.7014629	0.019429302	12567.72931	
	475	4959.220697	3559.772763	27.14724293	0.019429302	14134.60449	
Low.shot.size.setting	90	56.09887804	3328.067141	527.4555181	0.01926126	3040.827175	
	145	92.21568414	3351.097923	391.135222	0.019278527	3699.206644	
	170	115.5201921	3341.946078	339.2900627	0.019271681	4018.547115	
	195	144.6760704	3328.182988	293.5623428	0.019261347	4354.261705	

	215	173.3634721	3324.784812	261.1560179	0.019258789	4637.535292
	235	207.408275	3311.783113	232.0673372	0.019248973	4933.716765
	275	297.3027842	3315.138252	183.0886229	0.01925151	5579.143965
	285	325.3256387	3322.726473	172.5672153	0.019257238	5753.750217
	325	466.1145347	3326.799679	135.6371795	0.019260306	6482.126127
	343	547.8898827	3346.19524	121.8244352	0.019274862	6845.873024
	360	636.3393016	3379.392242	110.1681634	0.019299563	7214.356477
	375	730.1972041	3443.764837	101.1297412	0.019346714	7579.667233
	445	1354.986578	3559.772763	66.84301323	0.019429302	9406.60238
	475	1769.642922	3559.772763	55.57988874	0.019429302	10245.99654
	490	2025.009158	3559.772763	50.62500166	0.019429302	10681.47142
high.melt.temperature	90	61.97973146	1781.358652	381.4684535	0.072674684	2430.693615
	145	101.8209457	1788.011979	287.6259476	0.072874412	3006.598403
	170	127.5232468	1778.699701	250.9229013	0.072874395	3284.762083
	195	159.7020555	1773.091339	218.4524253	0.072874394	3581.267385
	215	191.2173179	1770.045546	195.2034195	0.072874374	3831.24835
	235	228.9178284	1767.327516	174.1874678	0.072874383	4093.00991
	275	328.1726592	1769.705783	138.3280611	0.072874349	4658.887177
	285	359.0836679	1769.705783	130.4689328	0.01763134	4808.011539
	325	514.5105821	1773.108571	103.0776142	0.017636448	5444.635101
	343	604.7097525	1775.999346	92.63763233	0.017640779	5753.717185
	360	704.6266604	1786.443126	83.79981428	0.01765636	6065.271397
	375	806.23873	1807.28163	76.84114995	0.017687154	6365.474652
	445	1497.616744	1841.281371	50.76984776	0.017736571	7896.746764
	460	1710.840927	1739.572906	45.52565941	0.017585635	8104.5417
	475	1955.921124	1739.572906	41.53300644	0.017585635	8462.434896
	490	2238.168017	1739.572906	37.85871108	0.017585635	8826.716072
	505	2557.265219	1739.572906	34.53892801	0.017585635	9218.713869
Low.melt.temperature	90	46.24184236	5491.957049	723.3749162	0.020487101	3429.618317
	145	94.23679148	5518.894281	463.2713533	0.020498799	4465.29058
	170	130.2968469	5563.449069	376.3403568	0.020518012	5007.893018
	195	179.928979	5548.187127	303.8236824	0.02051145	5581.563164
	215	232.9170361	5530.37255	255.4457204	0.020503765	6074.084484
	235	301.4470246	5517.990746	214.564553	0.020498408	6603.712378
	260	416.1804642	5511.903751	172.3248116	0.020495769	7322.138967
	275	505.0812691	5517.138349	151.0375693	0.020498039	7787.734083
	285	574.7243259	5530.028524	138.3530942	0.020503616	8115.943043
	315	846.7121186	5548.571759	106.0980297	0.020511615	9164.935177
	325	963.0517147	5541.831613	97.03869404	0.020508711	9536.543849
	343	1214.503827	5555.417017	82.71246169	0.020514561	10253.21802
	360	1511.847781	5596.16634	71.21714113	0.020532013	10992.95781
	375	1833.830013	5704.901405	62.6423762	0.020577913	11733.67976
	460	5399.18406	5319.815466	28.87435675	0.020410835	16191.31884
	475	6539.481329	5319.815466	25.25181828	0.020410835	17182.98238
	490	7926.678385	5319.815466	22.07001672	0.020410835	18224.03471
	525	12447.6334	5319.815466	16.08676818	0.020410835	20863.85874

High.piston.speed.setting	Flowlength.(mm)	Unmodfied.Fractional.Coverage	Adjusted.Fractional.Coverage
	90	0.664455133	0.622486
	145	0660600195	0.615111
	170	0.659903584	0.613844
	195	0.653815611	0.607622
	215	0.643863879	0.597584
	260	0.643420578	0.596501
	275	0.6420786	0.594812
	285	0.641032095	0.593496
	315	0.63	0.585567
	325	0.6320329	0.589089
	343	0.630065371	0.589577
	360	0.6241699	0.583814
	375	0.618633233	0.582766
	460	0.527927299	0.527927
	475	0.511768978	0.511769
	490	0.498364331	0.498364
Low.piston.speed.setting	90	0.689057396	0.642787
	145	0.692621509	0.641378
	170	0.692304866	0.640101
	195	0.689900398	0.637213
	215	0.68720811	0.634255
	235	0.68546886	0.631825
	260	0.685555	0.631767
	275	0.684666754	0.630608
	285	0.687869965	0.633622
	315	0.680615113	0.626422
	325	0.688518593	0.634829
	343	0.669690715	0.616585
	360	0.667162285	0.614259
	375	0.66521013	0.61333
	415	0.643249142	0.637764
	445	0.562980759	0.541557
	460	0.533264801	0.515547
	475	0.512741854	0.497234
	490	0.505587609	0.490481
Base.Setting	90	0.672333745	0.628863
	145	0.680039601	0.632394
	170	0.667376696	0.618995
	195	0.668113545	0.619465
	215	0.669887703	0.621097
	235	0.665519991	0.616192
	260	0.66042195	0.610985
	275	0.663392543	0.613673
	285	0.65819495	0.608263
	325	0.651606294	0.6021
	343	0.642900574	0.593893
	360	0.641887469	0.593031
	375	0.635561537	0.587596
	445	0.536369509	0.514946
	460	0.520796145	0.503078
	475	0.51029797	0.49479
	490	0.492728249	0.477621
High.delay.time.setting	90	0.71303761	0.657013
	145	0.6994871	0.639556
	170	0.701769188	0.641248
	195	0.699996603	0.639138
	215	0.701257595	0.641201
	235	0.698408593	0.636539
	275	0.699423999	0.637049
	285	0.698485672	0.63587
	315	0.699962583	0.637029
	325	0.699467845	0.636939
	343	0.686372017	0.624064
	360	0.680490058	0.617831
	375	0.670616312	0.608574
	445	0.54996579	0.528542
	460	0.516916765	0.499199
	475	0.498031168	0.482523
	490	0.493316219	0.478209
	525	0.486413845	0.474701
Low.delay.time.setting	90	0.646852772	0.611553
	145	0.641527395	0.602575
	170	0.6402977	0.600702
	195	0.642550577	0.602863
	215	0.641688346	0.601948
	235	0.640829917	0.60076
	275	0.634794885	0.594288
	285	0.629447227	0.588688
	325	0.622278177	0.581826
	343	0.619749266	0.5797631
	360	0.613333547	0.573584
	375	0.602327828	0.56339
	445	0.541584269	0.52016
	460	0.527157593	0.50944
	475	0.518919719	0.503412
High.gas.pressure.setting	90	0.675408624	0.632121
	145	0.668449729	0.620965
	170	0.665198697	0.616856
	195	0.666624644	0.617826
	215	0.661579087	0.612437
	235	0.658701847	0.608766
	275	0.659627084	0.608661
	285	0.658785575	0.607419
	315	0.65831585	0.606272
	343	0.648579795	0.596725
	360	0.642704076	0.590414
	375	0.639065328	0.587139
	445	0.568488481	0.576515
	460	0.537996416	0.537996
	475	0.522383791	0.522384
	490	0.512449143	0.512449
Low.gas.pressure.setting	90	0.698873255	0.653428
	145	0.692935765	0.643289
	170	0.690407814	0.640018
	195	0.691673269	0.640974
	215	0.688984876	0.63811
	235	0.692774128	0.641291
	260	0.694173317	0.642539
	275	0.688442735	0.6365
	325	0.678472635	0.626616
	343	0.669053441	0.617624
	360	0.661023285	0.609632
	375	0.658290052	0.607719
	445	0.586277093	0.564853
	460	0.526014283	0.508296

	475	0.515202825	0.499695
	490	0.496022157	0.480915
High shot size setting	90	0.679814469	0.635332
	145	0.678385666	0.629732
	170	0.678443498	0.629057
	195	0.675245682	0.625578
	215	0.675144517	0.625324
	235	0.678017961	0.627631
	275	0.670001147	0.619199
	285	0.671366624	0.620347
	325	0.669771592	0.619135
	343	0.659192099	0.609024
	360	0.656571262	0.606505
	375	0.650316108	0.601111
	445	0.593335238	0.571911
	475	0.547726996	0.547727
Low shot size setting	90	0.673297245	0.631405
	145	0.672809622	0.627065
	170	0.669223792	0.622878
	195	0.668933217	0.622489
	215	0.667955017	0.619809
	235	0.66403757	0.617227
	275	0.662242917	0.615314
	285	0.659374749	0.612302
	325	0.657487586	0.611169
	343	0.654273822	0.608636
	360	0.636189927	0.590942
	375	0.625708532	0.581522
	445	0.500466487	0.479042
	475	0.473828363	0.458321
	490	0.457995038	0.442888
High.melt.temperature	90	0.693084074	0.651943
	145	0.690318652	0.646481
	170	0.687879622	0.643883
	195	0.68319826	0.63918
	215	0.684136001	0.639926
	235	0.679103836	0.634984
	275	0.679561568	0.635104
	285	0.67289133	0.628404
	325	0.6701	0.626261
	343	0.66521	0.622266
	360	0.65222	0.60943
	375	0.6294	0.587162
	445	0.575210798	0.553787
	460	0.539712115	0.521994
	475	0.529919918	0.514412
	490	0.527618377	0.512511
Low.melt.temperature	505	0.515450591	0.503736
	90	0.686706115	0.640392
	145	0.679323206	0.628394
	170	0.684293066	0.630512
	195	0.689236851	0.634989
	215	0.684060921	0.629586
	235	0.686483024	0.632174
	260	0.681287794	0.626932
	275	0.677176137	0.622631
	285	0.672125212	0.617256
	315	0.668292035	0.612535
	325	0.667041945	0.611765
	343	0.665278969	0.610409
	360	0.661918419	0.607622
	375	0.65875516	0.605085
	460	0.562243403	0.540819
	475	0.544177804	0.52646
	490	0.532056223	0.516548
	525	0.511946383	0.496839